Journal of Artificial Intelligence Research 1 (1993) 1-15 Submitted 6/91; published 9/91

Resource-bounded Norm Monitoring in Multi-agent Systems

Natalia Criado NATALIA.CRIADO@KCL.AC.UK
King's College London
Bush House, WC2B 4BG, London, UK

Abstract

Norms allow system designers to specify the desired behaviour of a sociotechnical system. In this way, norms regulate what the social and technical agents in a sociotechnical system should (not) do. In this context, a vitally important question is the development of mechanisms for monitoring whether these agents comply with norms. Proposals on norm monitoring often assume that monitoring has no costs and/or that monitors have unlimited resources to observe the environment and the actions performed by agents. In this paper, we challenge this assumption and propose the first practical resource-bounded norm monitor. Our monitor is capable of selecting the resources to be deployed and use them to check norm compliance with incomplete information about the actions performed and the state of the world. We formally demonstrate the correctness and soundness of our norm monitor and study its complexity. We also demonstrate in randomised simulations and benchmark experiments that our monitor can select monitored resources effectively and efficiently, detecting more norm violations and fulfilments than other tractable optimization approaches and obtaining slightly worse results than intractable optimal approaches.

1. Introduction

Norms allow system designers to specify the desired behaviour of a sociotechnical system (Jones et al., 2013; Singh, 2013). As an illustrative example, we can think of a smart home as a sociotechnical system. Assuming that initially the smart home allows users to perform actions freely, governance norms overriding this default rule and defining which actions cannot/must be done can be easily defined using prohibitions and obligations. For example, in this environment a reasonable norm can be to forbid users from opening any window if the air conditioner is functioning. Regimenting norms in sociotechnical systems (e.g., locking windows when the air conditioner is on) may not only be undesirable (e.g., users can get frustrated if they cannot open the window) but also insecure (i.e., it is unrealistic to assume that the norm designer has considered all the potential exceptions to a norm). Hence, it is impossible to prove that the behaviours of the users in the smart home will abide by the norms, since behaviour cannot be predicted a priori, which makes impossible the application of traditional verification techniques (Keller, 1976) such as model checking (Günay et al., 2016) and testing (Winikoff & Cranefield, 2014). In such complex systems, it is more desirable to allow norm violations to happen and to implement norm enforcement mechanisms.

Norm enforcement mechanisms (Grossi et al., 2007) persuade agents (e.g., users in our smart home scenario) to comply with norms by signalling norm violations and fulfilments and applying sanctions and rewards, respectively. A key element of such mechanisms is norm monitoring; i.e., the process by which agent's actions are observed and checked against the norms. Norm monitoring requires the deployment of different types of resources (i.e., sensors and controllers such as motion sensors, temperature controls, window and door lock sensors) that control the execution of actions and also sense properties of the environment (e.g., temperature controls allow users to turn

on/off the air conditioner and sense the temperature). The actions and properties observed through deployed resources are used by norm monitors to check norm compliance. However, resources can be expensive and usually there is a limited budget that can be spent on monitoring norm compliance (Criado, 2017). Hence, an important question is how to determine which resources need to be deployed to check compliance with a set of norms given a budget.

In this paper, we propose the first practical norm monitor that selects the resources to be deployed based on their cost and value to norm monitoring. The paper contributions can be summarised as follows: (i) for the first time, we propose a method to monitor norm compliance online under incompleteness demonstrating its correctness and soundness; (ii) we propose a method to analyse offline the guarantees offered by our norm monitor; (iii) we propose a tractable algorithm to select the resources to be deployed based on their cost and value to norm monitoring (their capabilities to detect norm violations and fulfilments); and (iv) we experimentally demonstrate that our resource selection mechanism obtains solutions near optimal and surpasses other tractable optimization approaches.

This paper is organized as follows. Section 2 contains the preliminary definitions used in this paper. Section 3 defines our resource-bounded norm monitor and describes its information model. Section 4 contains the algorithms executed by our norm monitor. Our proposal is experimentally evaluated in Section 5. Related work is discussed in Section 6. Finally, conclusions are contained in Section 7.

2. Preliminaries

We make use of a finite set of propositional symbols, denoted by P, that characterise the properties of the world relevant to norm monitoring. Given a proposition p, we denote by $lit(p)$ the literals that can be built using p; i.e., the set formed by p and its negation $\neg p$. We extend this definition to sets of propositions $\mathcal{P} = \{p_1, ..., p_n\}$ as follows: $lit(\mathcal{P}) = \bigcup lit(p_i)$. Similarly, given a literal l we denote by $prop(l)$ the proposition contained in the literal. We also make use of the true (vs. false) proposition $\top$ (vs. $\bot$).

By a state, we mean the properties of the world that are true at a particular moment; i.e., a state is built on a "*closed world assumption*" and defined by a set of properties $s \subseteq P$ that hold at a given moment. A "*closed world assumption*" (Reiter, 1978) establishes that states are complete and any a property p that holds in a state s is also known to hold in that state; i.e., $s \vdash p$ iff $p \in s$, and $s \nvdash p$ iff $p \notin s$.

We define that a state $s \subseteq P$ satisfies a condition represented as a literal $l \in lit(P)$, denoted by $s \vDash l$, iff $l \in s \vee \bar{l} \notin s$, where $\bar{l}$ denotes the negation of literal l. We extend this definition to sets of literals $c \subseteq lit(P)$; i.e., $s \vDash c$, iff $\forall l \in c : l \in s \vee \bar{l} \notin s$. We also give the common semantics to the true (vs. false) proposition; i.e., for any state s, $s \vDash \top$ (vs. $s \nvDash \bot$)

***Example* 1.** *In our smart home scenario let us assume that the finite set of propositional symbols representing the state of the world is defined as:*

$$P = \{openWindow, onAC, tUp, tDown\}$$

where: $openWindow$ represents that the window is open, $onAC$ represents that the air conditioner is on, and tUp and $tDown$ represent that the temperature of the thermostat has been increased or decreased, respectively. For example, a possible initial state s_0 could be defined as follows:

$$s_0 = \emptyset$$

Note that by the closed world assumption propositions $onAC, tUp, tDown$ and $openWindow$ are false in the initial state.

2.1 Action Definition

In line with the existing literature (Boutilier & Brafman, 2001), actions are represented using pre-conditions and postconditions[1]. If a state satisfies the precondition, then the action can be executed transforming the current state into a new state in which the postcondition holds; i.e., the propositions in the negative (vs. positive) literals appearing in the postcondition are deleted from (vs. added to) the state.

Definition 1 (Action)**.** *An action is a tuple $\langle pre, post\rangle$ where:*

- *$pre \subseteq lit(P)$ is a set of literals representing the action precondition;*
- *$post \subseteq lit(P)$ is a set of literals representing the action postcondition.*

Given an action a, we denote by $pre(a)$ and $post(a)$ the action precondition, and postcondition. Given a set of actions $\mathcal{A} = \{a_1, ...a_n\}$, we define $pre(\mathcal{A}) = \bigcup pre(a_i)$ and $post(\mathcal{A}) = \bigcup post(a_i)$.

We denote by A the set of actions that can be executed in a given scenario. We assume that these actions are deterministic and the state evolves due to action execution only. We assume that each agent performs at most one action at a time[2], but that more than one agent can perform an action at a given time. Note also that, by the definition of state, actions executed at a time are consistent; i.e., there are no contradictions among the preconditions and postconditions of actions executed simultaneously [3].

Example 2. *In the smart home scenario there are four actions to set up the thermostat temperature:*

$$increase : \langle\{onAC, \neg tDown\}, \{tUp\}\rangle$$

$$increaseToBaseline : \langle\{onAC, tDown\}, \{\neg tDown\}\rangle$$

$$decrease : \langle\{onAC, \neg tUp\}, \{tDown\}\rangle$$

$$decreaseToBaseline : \langle\{onAC, tUp\}, \{\neg tUp\}\rangle$$

the actions to open/close the window can be defined as follows:

$$open : \langle\{\neg openWindow\}, \{openWindow\}\rangle$$

$$close : \langle\{openWindow\}, \{\neg openWindow\}\rangle$$

and the actions to turn on/off the air conditioner can be defined as follows:

$$turnOn : \langle\{\neg onAC\}, \{onAC\}\rangle$$

$$turnOff : \langle\{onAC\}, \{\neg onAC\}\rangle$$

where $increase, decrease, increaseToBaseline, decreaseToBaseline, open, close, turnOn$ and $turnOff$ are just identifiers used in this paper to represent the actions more compactly.

1. Without loss of generality, we assume actions are instantaneous, but this can be relaxed by decomposing durative actions into sets of instantaneous actions.
2. This limitation can be relaxed by decomposing agents into groups of agents corresponding to agents' actuators (Boutilier & Brafman, 2001).
3. This is a realistic assumption as in many domains some of the inconsistent actions fail; e.g., a window cannot be open and closed simultaneously. Relaxing this assumption is not desirable, as this would entail that contradictory actions can be executed simultaneously leaving the world in an inconsistent state. Our monitor is agnostic about how this is achieved; e.g., by design, by coordinating agent actions, by having a fine time granularity, etc.

2.2 Resource Definition

A resource is formed by coupled controls and sensors that have a unique cost. Controls manage the execution of actions and therefore the execution of these actions by agents can be observed thought the resources, whereas sensors allow some properties of the world to be observed. For example, a climatic resource can be deployed (brought into effective action) to control the interaction of with the air conditioner (i.e., to observe the execution of actions to set up the temperature and activating the air conditioner) and to observe the thermostat. The deployment cost of a resource can represent the economic cost of installing and maintaining it, the cost associated with processing the sensor and control data obtained through the resource, etc.

Definition 2 (Resource)**.** *A resource is defined as a tuple* $\langle control, sensor, cost \rangle$*, where:*

- $control \subseteq A$ *is the set of actions whose execution is controlled by the resource and that can be observed if the resource is deployed;*
- $sensor \subseteq P$ *is the set of properties of the world that can be observed if the resource is deployed and, based on these observations, their truth value can be established;*
- $cost \in \mathbb{R}_{\geq 0}$ *is the cost of the resource.*

Given a resource r, we define $control(r)$, $sensor(r)$ and $cost(r)$ as the actions and properties that can be observed through the resource and the cost of the resource.

Definition 3 (Discernible Literals)**.** *Given a resource* r*, the set of discernible literals that are observed or can be inferred (i.e., deduced from the actions observed) through the resource is defined as follows:*

$$dis(r) = lit(sensor(r)) \bigcup \left(\bigcup_{\forall a \in control(r)} pre(a) \cup post(a) \right) \tag{1}$$

Again, we extend $control, sensor, cost$ and dis definitions to sets of resources.

***Example* 3.** *Let us assume the existence of two different resources that can be deployed:*

- *A resource controlling the interaction with the air conditioner allowing users to turn it on/off and to change the temperature:*

  $$r_1 : \langle \{turnOn, turnOff, increase, decrease, increaseToBaseline, decreaseToBaseline, \{onAC, tUp, tDown\}, 5 \rangle$$

  The literals discernible through resource r_1 *are as follows:*

  $$dis(r_1) = \{onAC, tUp, tDown, \neg onAC, \neg tUp, \neg tDown\}$$

- *A resource controlling the interaction with the window:*

  $$r_2 : \langle \{open, close\}, \{openWindow\}, 1 \rangle$$

  The literals discernible through resource r_2 *are follows:*

  $$dis(r_2) = \{openWindow, \neg openWindow\}$$

Again r_1 *and* r_2 *are resource identifiers.*

2.3 Norm Definition

The purpose of this paper is not to propose, compare or improve existing normative definitions, but to make use of a well-known definition in proposing our norm monitor.

As in the related literature (Alechina et al., 2013; Lorini, 2012; Alechina et al., 2014), we consider a "*closed legal system*", where by default agents are permitted to do (vs. not to do) any action, and obligation and prohibition norms define exceptions to this default rule. In particular, in this paper we consider norms as formal statements that define patterns of behaviour by means of deontic modalities; i.e., obligations and prohibitions (López y López et al., 2006; Vasconcelos et al., 2007). These patterns of behaviour determine under which circumstances agents are obliged to or forbidden from performing certain actions. The circumstances in which a norm becomes relevant and must be observed are defined in terms of activation and expiration conditions as follows:

Definition 4 (Norm)**.** *A norm is defined as a tuple* $\langle deon, target, activ, expir \rangle$*, where:*

- $deon \in \{\mathcal{O}, \mathcal{F}\}$ *is the deontic modality, determining if the norm is an obligation (*$\mathcal{O}$*) or prohibition (*$\mathcal{F}$*);*
- $target \in A$ *is the action whose execution is regulated by the norm, i.e., the mandatory or forbidden action;*
- $activ$ *is a set such that* $activ \subseteq lit(P)$ *or* $activ = \{\top\}$ *or* $activ = \{\bot\}$ *representing the activation condition;*
- $expir$ *is a set such that* $expir \subseteq lit(P)$ *or* $expir = \{\top\}$ *or* $expir = \{\bot\}$ *set of representing the expiration condition.*

Given a norm n, we define $deon(n)$, $target(n)$, $activ(n)$ and $expir(n)$ as the deontic modality, the norm target, and the activation and expiration conditions, respectively. Again we extend these definitions to sets of norms.

We define that a norm (n) becomes relevant in a specific state (s) when the activation condition is satisfied by the state ($s \vDash activ(n)$). Similarly, we define that a relevant norm expires when the expiration condition is satisfied by a state ($s \vDash expir(n)$). Thus, the expiration condition defines deadline conditions modelled as propositions (Dignum et al., 2004). According to this definition, in those states where both the activation and expiration condition of a norm are satisfied simultaneously, the norm is considered as not relevant. This norm definition has been extensively used in the related literature (Meneguzzi et al., 2012; Criado & Such, 2016; Oren et al., 2009; Kollingbaum & Norman, 2002; Beheshti et al., 2015) to represent norms in a wide range of problems and domains.

The semantics of norms depends on their deontic modality. In accordance with the standard semantics of deontic logics (von Wright, 1981)[4], a relevant obligation norm is fulfilled when the action in the norm target is performed before it expires and violated otherwise, while a relevant prohibition norm is violated when the action in the norm target is performed before it expires and fulfilled otherwise.

4. In consonance with von Wright's proposal, we do not represent the norm subject explicitly and, without loss of generality, we consider that norms are addressed to all agents. However, our norm definition and norm monitoring algorithms could be trivially extended to represent and consider the norm subject; i.e., the agent or agents responsible for complying with a norm.

***Example* 4.** *In our example, let us assume that the following three norms have been defined:*

- *Two unconditional norms, which are always relevant, forbidding users from increasing and decreasing the temperature of the thermostat (which has been previously set to an optimal temperature). This can be formalised as follows:*

$$n_1 : \langle \mathcal{F}, increase, \{\top\}, \{\bot\} \rangle$$

$$n_2 : \langle \mathcal{F}, decrease, \{\top\}, \{\bot\} \rangle$$

  The activation and expiration conditions are defined as $\{\top\}$ and $\{\bot\}$, respectively. In this case, the norm is an unconditional norm which is always relevant since the norm is activated in all states and there is no state in which the norm expires.

- *A conditional norm that forbids users from opening the window when the air conditioner control is on. This is represented as follows:*

$$n_3 : \langle \mathcal{F}, open, \{onAC\}, \{\neg onAC\} \rangle$$

  This norm becomes relevant when the air conditioner is on and it expires when the air conditioner is off.

Again n_1, n_2 and n_3 are norm identifiers used for compactness.

3. Norm Monitor

In this section we define our resource-bounded norm monitor and describe its information model. The definition of norm monitor and the norm monitoring process have been adapted from our previous work (Criado & Such, 2016, 2017). In particular, that work proposes an information model to monitor norm compliance under incomplete observations by predicting future actions (Criado & Such, 2016) and reconstructing unobserved past actions (Criado & Such, 2017). In this section we expand this information model with a method to select the resources deployed and also demonstrate the soundness and correctness of the norm monitoring process.

Definition 5 (Norm Monitor)**.** *A norm monitor is defined as a tuple $\langle R, N, b \rangle$ where:*

- *R is a set of resources that can be deployed;*
- *N is the set of norms that regulate agents' actions;*
- *$b \in \mathbb{R}_{\geq 0}$ represents the budget of the norm monitor.*

The norm monitor has to solve two closely related problems: (i) how to use the actions and properties observed through the deployed resources to monitor norm compliance; and (ii) how to select a set of resources to be deployed satisfying the budget condition such that the detection of norm violations and fulfilments is maximal. The solutions to these two problems are detailed below[5].

Norm enforcement is out of the scope of this work and has been covered extensively in the related literature (Meneguzzi et al., 2012; Fornara & Colombetti, 2008b). Hence, we assume that once the monitor detects a norm violation (vs. fulfilment), it applies the corresponding sanction (vs. reward).

5. Note that the solution to these two problems will vary according to the normative definition used, but the problems will be solved in a similar way. For instance, Appendix A illustrates how our monitor can deal with maintenance and achievement norms defined over properties of the world, and with conjunction norms; whereas Appendix B shows how our norm monitor can accommodate norm importance.

***Example* 5.** *In our example, there is a monitor defined as follows:*

$$\langle \{r_1, r_2\}, \{n_1, n_2, n_3\}, 5 \rangle$$

For the time being, assume that the smart home owner has decided to deploy resource r_2, since it has a lower cost.

3.1 Norm Monitoring Process

As aforementioned, the monitor has partial information about the state of the world. The monitor represents the knowledge it has about the state of the word using an "*open world assumption*" as a set of literals that are known in a state. An "*open world assumption*" (Reiter, 1978) determines that the monitor's knowledge about states are incomplete and the truth value of a proposition may be true irrespective of whether or not it is known to be true.

We denote by a partial state, represented by $\widehat{s}$, the norm monitor's knowledge about the state of the world. A partial state contains positive (vs. negative) literals representing properties known to be true (vs. false) in the state. By the open world assumption $p \in \widehat{s}$ if $s \vdash p$ and $\neg p \in \widehat{s}$ if $s \nvdash p$.The rest of properties are unknown. Partial states also contain literals corresponding to the true and false propositions; i.e., $\forall \widehat{s} : \top \in \widehat{s}$ and $\neg\bot \in \widehat{s}$.

We assume that the monitor has complete knowledge of the initial state. Thus, at $t = 0$ the monitor knows which literals are true or false in the current state, i.e., the *partial* state $\widehat{s}_0$ is equivalent to s_0 (i.e., for all proposition $p \in s_0$ it follows that $p \in \widehat{s}_0$, and for all proposition $p \in P \setminus s_0$ it follows that $\neg p \in \widehat{s}_0$). From that moment on the actions executed by agents change the state of the world; meaning that the actions executed at time t (denoted by Act_t) evolve s_t into s_{t+1} and, as a consequence, the monitor's knowledge $\widehat{s}_t$ also evolves into $\widehat{s}_{t+1}$. Recall that actions executed at a time are consistent; i.e., in Act_t there are no contradictions among the action preconditions ($\nexists l \in pre(Act_t) : l, \overline{l} \in pre(Act_t)$) and postconditions ($\nexists l \in post(Act_t) : l, \overline{l} \in post(Act_t)$), so the monitor is aware of the fact that the states s_t and s_{t+1} are consistent when it updates its knowledge (i.e., the partial states $\widehat{s}_t$ and $\widehat{s}_{t+1}$).

***Example* 6.** *In our example, the monitor knows which propositional formulas are true or false in the initial state and $\widehat{s}_0$ is defined as follows:*

$$\widehat{s}_0 = \{\neg onAC, \neg tUp, \neg tDown, \neg openWindow, \top, \neg\bot\}$$

3.1.1 STATE UPDATE

The monitor uses the resources deployed (denoted by $S \subseteq R$) to observe the actions and the properties of the world. We assume observations may be incomplete, but are accurate (not noisy). At time t the monitor observes some of the actions performed by agents and the truth value of some of the properties describing the state of the world. In particular, the actions observed by the monitor (denoted by $\widehat{Act_t}$) are defined as those executed actions that are controllable by selected resources:

$$\widehat{Act_t} \leftarrow Act_t \cap control(S) \tag{2}$$

The actions observed are used to increase the knowledge about the current state. In particular, the current partial state $\widehat{s}_t$ is updated considering the preconditions of observed actions $\widehat{Act_t}$ as follows:

$$\widehat{s}_t \leftarrow \widehat{s}_t \bigcup pre(\widehat{Act_t}) \tag{3}$$

Similarly, the truth values of the properties that can be observed can be used to define the new resulting partial state (i.e., the following equation adds literals corresponding to those properties that hold and do not hold in the next state and can be sensed):

$$\widehat{s}_{t+1} \leftarrow (sensor(S) \cap s_{t+1}) \cup \{\neg p | p \in (sensor(S) \cap (P \setminus s_{t+1}))\} \tag{4}$$

As $\widehat{Act}_t \subseteq Act_t$, the monitor cannot be sure about the effects of non-observable actions (i.e., actions whose execution is not controlled by any resource deployed). Thus, the new partial state $\widehat{s}_{t+1}$ is extended by: considering that the postconditions of the observed actions hold, determining which dynamic properties have not been modified by the observed and non-observable actions, and considering that the rest of dynamic propositions are unknown. In particular, the new partial state $\widehat{s}_{t+1}$ is updated considering the effect of actions $\widehat{Act}_t$ executed in $\widehat{s}_t$ as follows:

$$\widehat{s}_{t+1} \leftarrow \widehat{s}_{t+1} \cup post(\widehat{Act}_t) \ \cup inv(\widehat{s}_t) \tag{5}$$

where inv is formed by *invariant* literals; i.e., literals of $\widehat{s}_t$ that cannot be modified by any non-observable plausible action. An action is considered as plausible within the context of a partial state $\widehat{s}$ if the action could have been applied in state $\widehat{s}$. More formally, an action a is plausible within the context of a partial state $\widehat{s}$ iff $\nexists l \in pre(a) : \bar{l} \in \widehat{s}$. The invariant literals are defined as:

$$inv(\widehat{s}) = \bigcup_{\substack{\forall l \in \widehat{s} \wedge prop(l) \in : \mathcal{P} \\ \nexists a \in \{a | a \in (A \setminus control(S))\}: \\ \bar{l} \in post(a) \wedge (\nexists l' \in pre(a) : \bar{l'} \in \widehat{s})}} l \tag{6}$$

Note that the truth literals $\top$ and $\neg\bot$ are always invariants.

Theorem 1 (Correctness)**.** *The partial states generated by the norm monitor must be in accordance with the state of the world. More formally, partial states constructed at time t using equations 2,3, 4 and 7 are correct; i.e., any partial state $\widehat{s}_t$ must be satisfied by s_t , denoted by $s_t \vDash \widehat{s}_t$.*

Proof. We will demonstrate this by induction. We assume the monitor has complete knowledge of the initial state and we have that $s_0 \vDash \widehat{s}_0$. Then we assume that $s_{t-1} \vDash \widehat{s}_{t-1}$ and we demonstrate that $s_t \vDash \widehat{s}_t$ by contradiction. Let us assume $s_t \nvDash \widehat{s}_t$, thus there is at least one literal $l \in \widehat{s}_t$ such that $s_t \nvDash l$ (i.e., $s_t \vDash \bar{l}$).

1. If the proposition in l can be observed through S, then $\bar{l} \in \widehat{s}_t$ by Equation 4, which contradicts our premise.

2. If not, there are two different cases:

   2.1. The truth value of the proposition in l has changed due to an action executed at time $t-1$; i.e., $\exists a \in Act_{t-1} : \bar{l} \in post(a)$ and we have two different cases:

      2.1.1. If $a \in \widehat{Act}_{t-1}$, then $\bar{l} \in \widehat{s}_t$ since Equation 7 will add $post(\widehat{Act}_{t-1})$ to $\widehat{s}_t$, which contradicts the premise.

2.1.2. Else $a \notin \widehat{Act}_{t-1}$ and $a \in A \setminus control(S) : \bar{l} \in post(a) \wedge (\nexists l' \in pre(a) : \bar{l'} \in \widehat{s}_t)$, then Equation 7 cannot conclude that l is an invariant and $l, \neg l \notin \widehat{s}_t$, which also contradicts the premise.

2.2. The truth value of the proposition in l has not changed by any action executed at time $t-1$. In this case, $s_{t-1} \vDash \bar{l}$ and $l \notin \widehat{s}_{t-1}$ by the induction hypothesis.

2.2.1. If $\bar{l} \in \widehat{s}_{t-1}$ and $\nexists a \in A \setminus control(S) : l \in post(a) \wedge (\nexists l' \in pre(a) : \bar{l'} \in \widehat{s}_t)$, then Equation 7 will determine that $\bar{l}$ is an invariant and $\bar{l} \in \widehat{s}_t$ which contradicts our premise.

2.2.2. Else, Equation 7 cannot determine that $\bar{l}$ is an invariant and $l, \neg l \notin \widehat{s}_t$ which contradicts our premise.

□

Example 7. *In our example, let us assume that in the initial state action $turnOn$ is executed and the resulting state is:*

$$s_1 = \{onAC\}$$

However, the monitor does not observe the action $turnOn$, because this action is not controlled by a deployed resource (recall $S = \{r_2\}$); i.e., $\widehat{Act}_0 = \emptyset$. Thus, the monitor infers the next partial state as follows:

$$\widehat{s}_1 = \{\neg openWindow, \top, \neg\bot\}$$

Note that the monitor knows that the window is not open, as this literal is an invariant (i.e., the truth value of this proposition cannot be changed by a non-observable plausible action). Literals $\top$, $\neg\bot$ are also invariants. The truth value of the rest of the propositions is unknown.

3.1.2 NORM COMPLIANCE

The monitor analyses norm compliance both offline, determining compliance with which norms is guaranteed to be detected and to what degree; and online, by checking if agents' actions abide by the norms.

Offline Norm Compliance. Given the set S, the monitor determines the *guarantees* offered by this set with regard to norm monitoring; i.e., which norm violations or fulfilments can be detected using the resources deployed. In particular, it is possible that compliance with a norm is: (i) always detected —i.e., the satisfaction of its expiration/activation conditions is always known since the propositions in their literals are sensed through deployed resources and its target is also controlled by these resources; (ii) partially detected —i.e., the satisfaction of its expiration/activation conditions may eventually be known since some of their literals are discernible but not sensed and can only be inferred from actions controlled by deployed resources, whereas its target is controlled by deployed resources; or (iii) non-detected —e.g., when the norm target is not-observable.

Definition 6 (Perfectly Detectable Norm)**.** *We define that compliance with a norm $n \in N$ is* perfectly detectable *by a set of resources $S \subseteq R$ if it satisfies the following 3 properties:*

i) $target(n) \in control(S)$.

ii) $activ(n) \subseteq lit(sensor(S))$ *or* $activ(n) = \{\top\}$ *or* $activ(n) = \{\bot\}$.

iii) $expir(n) \subseteq lit(sensor(S))$ *or* $expir(n) = \{\top\}$ *or* $expir(n) = \{\bot\}$.

Definition 7 (Partially Detectable Norm)**.** *We define that compliance with a norm $n \in N$ is* partially detectable *by a set of resources $S \subseteq R$ if it satisfies the following 4 properties:*

i) *n is not perfectly detectable.*

ii) *$target(n) \in control(S)$.*

iii) *$activ(n) \subseteq dis(S)$ or $activ(n) = \{\top\}$ or $activ(n) = \{\bot\}$.*

iv) *$expir(n) \subseteq dis(S)$ or $expir(n) = \{\top\}$ or $expir(n) = \{\bot\}$.*

Definition 8 (Non-Detectable Norm)**.** *We define that compliance with a norm $n \in N$ is* non-detectable *by a set of resources $S \subseteq R$ when it is neither perfectly detectable nor partially detectable.*

***Example* 8.** *Given the resources selected ($S = \{r_2\}$), norms n_1 and n_2 are non-detectable since their targets are non-observable; whereas norm n_3 is also non-detectable since the truth value of its activation and expiration condition cannot determined with r_2.*

Online Norm Compliance. In order to determine norm compliance online, the monitor needs to represent the normative state; i.e., the information about the norms that are (vs. not) relevant at a particular state. To this aim, for each norm $n \in N$ we define a proposition α_n representing that norm n is relevant. Given that the monitor has partial knowledge about the state of the world, the monitor should control norms only when it is completely sure that the norms are relevant to ensure that the norm monitoring process is sound (e.g., the monitor cannot indicate that a violation has occurred when it has not in fact occurred). Without loss of generality, we assume that norms are not relevant in the initial state (i.e., $\forall n \in N : \neg\alpha_n \in \widehat{s}_0 \wedge s_0 \vDash \neg\alpha_n$). From that moment on, the monitor checks the current state to determine which norms are relevant in the next state.

Definition 9 (Relevance Update)**.** *Given a partial state $\widehat{s}_t$, the next partial state $\widehat{s}_{t+1}$, and a set of norms to be checked N; the next partial state is updated with the norm relevance information as follows:*

$$\widehat{s}_{t+1} \leftarrow \widehat{s}_{t+1} \cup relevanceUpdate(\widehat{s}_t, N)$$

where $relevanceUpdate$ is a function that updates the normative state:

$$relevanceUpdate(\widehat{s}, N) = \left(\bigcup_{\substack{\forall n \in N: \\ (activ(n) \subseteq \widehat{s} \vee \alpha_n \in \widehat{s}) \\ \wedge \exists l \in expir(n): \bar{l} \in \widehat{s}}} \alpha_n \right) \bigcup \left(\bigcup_{\substack{\forall n \in N: \\ expir(n) \subseteq \widehat{s}}} \neg\alpha_n \right)$$

A norm is known to be not relevant when it is known that the expiration condition was satisfied in the previous state. Similarly, a norm is known to be relevant when it is known that the norm has not expired since it became activated. Otherwise, the relevance of a norm is unknown[6].

Once the monitor has determined which norms are relevant, it checks compliance with these norms. To this aim, the monitor needs to represent which relevant obligation and prohibition norms

6. Note that by Equation 4, partial states are initialised in each time step, and in accordance with Equation 6 no literal relating to a proposition α_n can be considered as an invariant.

have been fulfilled and violated, respectively. Thus, for each norm $n \in N$, we define a proposition β_n denoting that the execution of the action in the norm target has been observed while it was relevant.

Definition 10 (Compliance Update)**.** *Given a partial state $\widehat{s}_t$, the next partial state $\widehat{s}_{t+1}$, the set of norms to be checked N, and the set of observed actions $\widehat{Act}_t$ at time t; the next partial state is updated with the norm compliance information as follows:*

$$\widehat{s}_{t+1} \leftarrow \widehat{s}_{t+1} \cup complianceUpdate(\widehat{s}_{t+1}, \widehat{s}_t, N, \widehat{Act}_t)$$

where $complianceUpdate$ is a function that updates the normative state:

$$complianceUpdate(\widehat{s}_{t+1}, \widehat{s}_t, N, \widehat{Act}_t) = \left(\bigcup_{\substack{\forall n \in N : \alpha_n \in \widehat{s}_{t+1} \wedge \\ (\neg\alpha_n \in \widehat{s}_t \vee \\ (\alpha_n \in \widehat{s}_t \wedge target(n) \notin \widehat{Act}_t \wedge \neg\beta_n \in \widehat{s}_t))}} \neg\beta_n \right) \bigcup \left(\bigcup_{\substack{\forall n \in N : \alpha_n \in \widehat{s}_{t+1} \wedge \alpha_n \in \widehat{s}_t \wedge \\ (target(n) \in \widehat{Act}_t \vee \beta_n \in \widehat{s}_t)}} \beta_n \right)$$

When the monitor knows that a norm has just become relevant, then it defines that the target has not been observed —i.e., if $\alpha_n \in \widehat{s}_{t+1}$ and $\neg\alpha_n \in \widehat{s}_t$, then literal $\neg\beta_n$ is added to the next partial state $\widehat{s}_{t+1}$. If the norm was already relevant in the previous time step, then the norm monitor checks the actions executed at time t to determine if the target has been observed. If so, it defines that the target has been observed —i.e., if $\alpha_n \in \widehat{s}_{t+1}$, $\alpha_n \in \widehat{s}_t$ and $target(n) \in \widehat{Act}_t$, then literal β_n is added to $\widehat{s}_{t+1}$. If the target was not observed at time t, then any information (if any) about the observance of the norm target is copied from the previous partial state —i.e., if $\alpha_n \in \widehat{s}_{t+1}$, $\alpha_n \in \widehat{s}_t$, then: literal β_n is added to $\widehat{s}_{t+1}$ iff $\beta_n \in \widehat{s}_t$ and $target(n) \notin \widehat{Act}_t$; or literal $\neg\beta_n$ is added to $\widehat{s}_{t+1}$ iff $\neg\beta_n \in \widehat{s}_t$.

Finally, the information about relevant norms, actions executed and norm compliance is taken into account to determine which norms have been violated and fulfilled.

Definition 11 (Norm Compliance Check)**.** *Given a partial state $\widehat{s}_t$, the previous partial state $\widehat{s}_{t-1}$, a norm n and a set of actions observed at time t ($\widehat{Act}_t$) , the norm is defined as:*

$$\begin{cases} fulfilled & \text{iff } \alpha_n \in \widehat{s}_t \text{ and } deon(n) = \mathcal{O} \text{ and } target(n) \in \widehat{Act}_t & \text{(10a)} \\ fulfilled & \text{iff } \neg\alpha_n \in \widehat{s}_t \text{ and } \alpha_n \in \widehat{s}_{t-1} \text{ and } deon(n) = \mathcal{F} \wedge \neg\beta_n \in \widehat{s}_{t-1} \text{ and } target(n) \in control(S) & \text{(10b)} \\ violated & \text{iff } \alpha_n \in \widehat{s}_t \text{ and } deon(n) = \mathcal{F} \text{ and } target(n) \in \widehat{Act}_t & \text{(10c)} \\ violated & \text{iff } \neg\alpha_n \in \widehat{s}_t \text{ and } \alpha_n \in \widehat{s}_{t-1} \text{ and } deon(n) = \mathcal{O} \wedge \neg\beta_n \in \widehat{s}_{t-1} \text{ and } target(n) \in control(S) & \text{(10d)} \\ unknown & \text{otherwise} \end{cases}$$

Just by observing the actions executed, the norm monitor may determine which relevant obligations are fulfilled and which prohibitions are violated. When the monitor knows that a norm has expired, then it may be able determine the violation of obligations and fulfilment of prohibitions based on the available information about the performance of the norm target. In particular, if an obligation norm expires and the target was not executed while it was relevant, then the obligation is violated. Similarly, when the target of a prohibition norm is not executed while the norm is relevant, then the

prohibition is fulfilled. Since the monitor only has partial knowledge of the actions executed, the fact that the target of a norm has not been observed does not mean that it has not been executed. Therefore, the monitor can only determine that obligations are violated and prohibitions are fulfilled when the execution of their targets is controlled by selected resources ($target(n) \in control(S)$).

Theorem 2 (Soundness)**.** *The norm compliance checking is sound; i.e., the norm monitor only detects violations (vs. fulfilments) when agents do not (vs. do) comply with norms.*

Proof. By exhaustion, we consider the fowling cases:

1. Definition 11 uses the same condition to detect the fulfilment of obligation norms and the violation of prohibition norms (corresponding to cases *(10a)* and *(10c)*, respectively). Hence, in our proof we consider a norm n and assume that at time t the current partial state is $\widehat{s}_t$, the monitor observes actions $\widehat{Act}_t$ and that case *(10a)* or *(10c)* applies (i.e., the monitor determines that n has been violated or fulfilled depending on its deontic modality), which entails that $target(n) \in \widehat{Act}_t$ and $\alpha_n \in \widehat{s}_t$. There are two main reasons why this detection could not be sound:

   1.1. The action $target(n)$ might have not been executed at time t ($target(n) \notin Act_t$), which is not possible since $\widehat{Act}_t \subseteq Act_t$ by Equation 2.

   1.2. The norm was not relevant in s_t. There are two reasons why a norm might not be relevant at time t:

      1.2.1. The norm has never become relevant; i.e., the activation condition was never satisfied. This entails that at some point in the past $t' < t$ the activation condition did not hold $\exists l \in activ(n) : s_{t'} \vDash \bar{l}$ but $l \in \widehat{s}_{t'}$ which according to Theorem 1 is not possible as $s_{t'} \vDash \widehat{s}_{t'}$.

      1.2.2. The norm was relevant but expired; i.e., at some point in the past $t' < t$ the expiration condition has been satisfied $s_{t'} \vDash expir(n)$ but $\exists l \in expir(n) : \bar{l} \in \widehat{s}_{t'}$ which is not possible since $s_{t'} \vDash \widehat{s}_{t'}$.

2. Definition 11 also uses the same condition to detect the fulfilment of prohibition norms and the violation of obligation norms (corresponding to cases *(10b)* and *(10d)*, respectively). Consider a norm n and assume that case *(10b)* or *(10d)* applies (i.e., the monitor determines that the norm has expired at time t and that it has been fulfilled or violated), which entails that: $\neg\alpha_n \in \widehat{s}_t$ and $\alpha_n \in \widehat{s}_{t-1}$, $target(n) \in control(S)$, and $\neg\beta_n \in \widehat{s}_{t-1}$. There are three main reasons why this detection could not be sound:

   2.1. The action $target(n)$ was executed at some point $t' < t$, but it was not observed ($target(n) \notin \widehat{Act}_{t'}$), which is not possible since $target(n) \in control(S)$ and $\widehat{Act}_{t'} = Act_{t'} \cap control(S)$ by Equation 2.

   2.2. The action $target(n)$ was executed and observed at some point $t' < t$ while the norm was relevant but the monitor did not know that the norm was relevant. In such case $\alpha_n \notin \widehat{s}'_t$ and by Definition 10 no information about the compliance with n would have been added to $\widehat{s}_{t'+1}$ making impossible that $\neg\beta_n \in \widehat{s}_{t-1}$, which contradicts our hypothesis.

   2.3. The norm has not expired. There are two reasons why a norm might not expire:

2.3.1. The norm was never relevant or the norm was relevant but expired in the past, which have already being demonstrated in case 1.2 not to be possible.

2.3.2. The norm still relevant, which is not possible as it entails that at some point $t' < t$ $\exists l \in expir(n) : s_{t'} \vDash \bar{l}$ and $l \in \widehat{s}_{t'}$ which according to Theorem 1 is not possible.

□

***Example* 9.** *The norm monitor uses the information about $\widehat{s}_0$ to update the new partial state with the normative information as follows:*

$$\widehat{s}_1 = \{\neg open, \alpha_{n_1}, \alpha_{n_2}, \neg\alpha_{n_3}, \top, \neg\bot\}$$

In particular, the norm monitor determines that the two unconditional norms are relevant adding α_{n_1} and α_{n_2} to the next partial state. The monitor also determines that norm n_3 is not relevant as its expiration condition was true in $\widehat{s}_0$.

No norm was relevant in the initial state and the monitor determines that the norms n_1 and n_2 have just become relevant in $t = 1$ updating $\widehat{s}_1$ with $\neg\beta_{n_1}$ and $\neg\beta_{n_2}$ to reflect that the targets of these norms have not been observed yet:

$$\widehat{s}_1 = \{\neg open, \alpha_{n_1}, \alpha_{n_2}, \neg\alpha_{n_3}, \neg\beta_{n_1}, \neg\beta_{n_2}, \top, \neg\bot\}$$

The monitor does not need to check norm compliance at $t = 0$ as no action was observed and no norm was relevant.

In the next time step ($t = 1$), action $increase$ is executed violating norm n_1. The monitor cannot observe this action through deployed resources ($\widehat{Act}_1 = \emptyset$) and the next partial state is defined as:

$$\widehat{s}_2 = \{\neg open, \alpha_{n_1}, \alpha_{n_2}, \neg\beta_{n_1}, \neg\beta_{n_2}, \top, \neg\bot\}$$

In this case, the monitor knows that norms n_1 and n_2 are still relevant and compliance information about them is copied from the previous partial state, whereas the relevance of norm n_3 is unknown. Again the monitor cannot check norm compliance as no action or norm expiration has been observed and the violation of n_1 is not detected.

In the next time step ($t = 2$), action $open$ is executed violating norm n_3. This action is observed through S ($\widehat{Act}_2 = \{open\}$) and the next state is defined as:

$$\widehat{s}_3 = \{open, \alpha_{n_1}, \alpha_{n_2}, \neg\beta_{n_1}, \neg\beta_{n_2}, \top, \neg\bot\}$$

However, the violation of n_3 is also undetected as the monitor does not know that norm n_3 was relevant at $t = 2$ (i.e., $\alpha_{n_3} \notin \widehat{s}_2$). In this example, the two norm violations are not detected due to the fact that deployed resources have not been properly selected. Next, we describe a method by which the norm monitor can select the resources deployed to maximize the detection of violations and fulfilments.

3.2 Selection of the Resources to be Deployed

The goal of the monitor is to select the resources to be deployed to maximize the detection of norm violations and fulfilments in accordance with the process defined the previous section. This is an optimization problem such as:

$$\max_{S \subseteq R} v(S) \textit{ subject to } cost(S) \leq b$$

where v is a function determining the value of a set of resources to norm monitoring.

3.2.1 RESOURCE VALUE

There are different ways in which the value of resources can be computed. In this paper we propose to calculate the value of a set of resources as an estimation of the number of norms that can be checked through these resources; i.e., the resources *coverage* of norm compliance as follows:

$$v(S) = \sum_{n \in N} cover(S, n)$$

where $cover(S, n)$ is the *coverage* of a norm provided by a set of resources, which measures the elements (i.e., actions and literals) belonging to a norm that can be observed through a set of resources. The norm monitoring process described in the previous section determines that to detect compliance with a norm its target must be controlled by deployed resources and its activation and expiration conditions must be discernible at least.

Definition 12 (Coverage)**.** *Given a norm $n \in N$, and a set of resources S; we define its coverage ($cover(S, n)$) as the ratio quantifying how many of the elements needed to determine compliance with n are discernible through or controlled by S as follows*[7]*:*

$$\frac{\nmid \{target(n)\} \cap control(S) \nmid + \nmid activ(n) \cap dis(S) \nmid + \nmid expir(n) \cap dis(S) \nmid}{1 + \nmid activ(n) \nmid + \nmid expir(n) \nmid}$$

where $\nmid$ is a modified version of the set cardinality defined as follows:

$$\nmid X \nmid = \begin{cases} 0 & \text{iff } X = \{\bot\} \text{ or } X = \{\top\} \\ |X| & \text{otherwise} \end{cases}$$

***Example* 10.** *Now let us assume that the monitor has the capability to select the resources to be deployed to maximize the detection of norm violations and fulfilments. In particular, the value of the different sets of resources is calculated as follows:*

$$v(\{r_1\}) = \frac{1}{1} + \frac{1}{1} + \frac{0+1+1}{3} = \frac{8}{3}$$

$$v(\{r_2\}) = \frac{0}{1} + \frac{0}{1} + \frac{1+0+0}{3} = \frac{1}{3}$$

$$v(\{r_1, r_2\}) = \frac{1}{1} + \frac{1}{1} + \frac{1+1+1}{3} = \frac{9}{3}$$

Obviously, the set of resources that offers a better coverage of the three norms is the set $\{r_1, r_2\}$. Deploying these two resources would allow the monitor to observe the target of all norms and the conditions under which these norms become relevant; i.e., with these resources all norms would be perfectly detectable. However, the cost of this set exceeds the monitor budget and the monitor selects the best alternative satisfying the budget condition, which is the set $\{r_1\}$. This resource allows the norm monitor to observe the target of the unconditional norms; i.e., two of the norms are perfectly detectable. Note that resource r_2 only offers a partial coverage of the conditional norm and no coverage of the two unconditional norms, being the resource with less potential to help the monitor to detect norm violations and fulfilments.

7. Our aim is not to analyse operations for information fusion, as this is done in (Bloch, 1996). We just make use of a ratio since it is well-known and efficient calculation satisfying submodularity.

If resource r_1 is selected, then when action $turnOn$ is executed at $t = 0$, the monitor would observe it ($\widehat{Act}_0 = \{turnOn\}$) and infer the next partial state as follows:

$$\widehat{s}_1 = \{onAC, \neg tDown, \neg tUp, \alpha_{n_1}, \alpha_{n_2}, \neg\alpha_{n_3}, \neg\beta_{n_1}, \neg\beta_{n_2}\}$$

In this case the monitor knows that the temperature has not changed, as literals $\neg tDown$ and $\neg tUp$ are invariants. The monitor does not need to check norm compliance since no norm is relevant at $t = 0$.

In the next time step ($t = 1$), action $increase$ is executed violating norm n_1. The monitor observes this action through r_1 ($\widehat{Act}_1 = \{increase\}$) and the next partial state is defined as:

$$\widehat{s}_2 = \{onAC, \neg tDown, tUp, \alpha_{n_1}, \alpha_{n_2}, \alpha_{n_3}, \beta_{n_1}, \neg\beta_{n_2}, \neg\beta_{n_3}\}$$

In this case, $onAC$ and $\neg tDown$ are invariants. When the monitor checks norm compliance it detects the violation of norm n_1 and registers that the action in its target has been observed while it was relevant adding β_{n_1} to $\widehat{s}_2$.

In the next time step ($t = 2$), action $open$ is executed violating norm n_3. The monitor does not observe this action through r_1 ($\widehat{Act}_2 = \emptyset$) and the next partial state is defined as:

$$\widehat{s}_3 = \{onAC, \neg tDown, tUp, \alpha_{n_1}, \alpha_{n_2}, \alpha_{n_3}, \beta_{n_1}, \neg\beta_{n_2}, \neg\beta_{n_3}\}$$

In this case, $onAC, \neg tDown$ and tUp are invariants. The violation of norm n_3 is not detected as the action in the target cannot be observed. However, thanks to our resource selection method the monitor has been able to detect one of the two norm violations.

3.2.2 RESOURCE SELECTION

Finding the optimal solution to the resource selection problem is NP-hard[8] and infeasible for real world problems. However, the resource value function presents an interesting property that allows us to compute efficient suboptimal solutions with a given guarantee. In particular, this function is submodular; i.e., it exhibits a diminishing property: selecting a resource when few resources have been selected has more value (i.e., provides more information about norm violations and fulfilments) than selecting it after more resources have been selected. More formally, we can prove that:

Theorem 3 (Submodularity)**.** *For all sets of resources $\mathcal{A} \subseteq \mathcal{B} \subseteq R$ and resource $s \in R \setminus \mathcal{B}$, it holds that:*

$$v(\mathcal{A} \cup \{s\}) - v(\mathcal{A}) \geq v(\mathcal{B} \cup \{s\}) - v(\mathcal{B})$$

Proof. Again we will consider the different cases.

1. Firstly, let us consider the case when $v(\mathcal{A} \cup \{s\}) - v(\mathcal{A}) = 0$; i.e., when adding s to $\mathcal{A}$ does not improve the value. This happens when there is not an element (action or discernible literal) in s such that this element is not in $\mathcal{A}$ and it is contained by a norm. More formally, $\nexists e : (e \in control(s) \vee e \in dis(s)) \wedge e \notin control(\mathcal{A}) \wedge e \notin dis(\mathcal{A}) \wedge (e \in activ(N) \vee e \in expir(N) \vee e \in target(N))$. Since $\mathcal{A} \subseteq \mathcal{B}$, it is obvious to demonstrate that $\nexists e : (e \in control(s) \vee e \in dis(s)) \wedge e \notin control(\mathcal{B}) \wedge e \notin dis(\mathcal{B}) \wedge (e \in activ(N) \vee e \in expir(N) \vee e \in target(N))$ and hence $v(\mathcal{B} \cup \{s\}) - v(\mathcal{B}) = 0$.

8. This problem can be modelled as a submodular maximization problem subject to knapsack constraints, which is NP-hard (Iyer & Bilmes, 2013).

2. Secondly, let us consider the case when $v(\mathcal{A} \cup \{s\}) - v(\mathcal{A}) > 0$; i.e., when adding s to $\mathcal{A}$ improves the value. This happens when the set $E = \{e | (e \in control(s) \vee e \in dis(s)) \wedge e \notin control(\mathcal{A}) \wedge e \notin dis(\mathcal{A}) \wedge (e \in activ(N) \vee e \in expir(N) \vee e \in target(N))\}$ is not empty. In this case $v(\mathcal{A} \cup \{s\}) - v(\mathcal{A}) > 0$. Since $\mathcal{A} \subseteq \mathcal{B}$, it is possible that all elements in E are contained in $\mathcal{B}$ and hence $v(\mathcal{B} \cup \{s\}) - v(\mathcal{B}) = 0$ which is lower than $v(\mathcal{A} \cup \{s\}) - v(\mathcal{A})$. The other extreme situation is that all the elements in E are not contained in $\mathcal{B}$, in this case it is trivial to demonstrate that $v(\mathcal{A} \cup \{s\}) - v(\mathcal{A}) = v(\mathcal{B} \cup \{s\}) - v(\mathcal{B})$. In any other case (where some elements of E are in $\mathcal{B}$) $v(\mathcal{A} \cup \{s\}) - v(\mathcal{A}) > v(\mathcal{B} \cup \{s\}) - v(\mathcal{B})$.

□

In the following section, we describe the algorithm executed by our norm monitor, including a greedy function for selecting deployed resources that leverages this submodularity property.

4. Norm Monitoring Algorithms

Algorithm 1 contains the pseudocode executed by the norm monitor. The monitor performs an initialisation step to select the resources to be deployed given a budget b (line 1). In each time step, the monitor uses the selected resources to observe the actions executed and properties of the world (lines 5 and 7), updates the knowledge about the state of the world (lines 5-9), and checks for each norm which ones have been violated and fulfilled (lines 10-25). Function OBSERVEACTIONS returns executed actions as defined by Equation 2 and Function OBSERVEPROPOSITIONS returns the truth value of sensed propositions as defined by Equation 4. The asymptotic cost of the norm monitoring process (corresponding to the while process in lines 4-26) is given by the cost of checking for each norm if it is relevant (line 7) and if its target has been executed (line 8). More formally, the cost is given by $O(|N| \times |P|)$ or $O(|N| \times |A|)$, whichever is bigger. The resource selection algorithm and its cost is described below.

Function 2 contains the pseudocode for the resource selection function. In particular, our monitor uses the CELF algorithm (Leskovec et al., 2007) to select deployed resources. The CELF algorithm performs: (i) a greedy search that uses the value to rank resources ignoring costs (lines 2-5) —i.e., the resource with the highest value is selected and added to the solution, this process is followed until it is not possible to add more resources without exceeding the budget; and (ii) a greedy search that uses the ratio value by cost to rank resources (lines 6-9) —i.e., the resource with the highest ratio value by cost is selected and added to the solution, this process is followed until it is not possible to add more resources without exceeding the budget. Then, the algorithm returns the solution with the highest value (line 10). This algorithm is known to obtain good approximate solutions to submodular maximization problems. In particular, it has been demonstrated that the CELF algorithm achieves an approximation guarantee of $\frac{1}{2}(1 - \frac{1}{e})$, whereas the empirical results show that this bound is much tighter. The asymptotic cost the algorithm is $O(|R|^2 \times |N| \times |P| \times |A|)$. Note, however, that the submodular property can be exploited to reduce the number of resource valuations needed.

Algorithm 1 Norm Monitoring Algorithm

Require: R, N, b, s_0
$S \leftarrow$ RESOURCESELECTION(R, N, b)
$t \leftarrow 0$
$\widehat{s}_{-1} \leftarrow \emptyset$
while $true$ **do**
  $\widehat{Act}_t \leftarrow$ OBSERVEACTIONS(S)
  $\widehat{s}_t \leftarrow \widehat{s}_t \cup pre(\widehat{Act}_t)$
  $\widehat{s}_{t+1} \leftarrow$ OBSERVEPROPOSITIONS$(S) \cup post(\widehat{Act}_t) \cup inv(\widehat{s}_t)$
  $\widehat{s}_{t+1} \leftarrow \widehat{s}_{t+1} \cup relevanceUpdate(\widehat{s}_t, N)$
  $\widehat{s}_{t+1} \leftarrow \widehat{s}_{t+1} \cup complianceUpdate(\widehat{s}_{t+1}, \widehat{s}_t, N, \widehat{Act}_t)$
  for all $n \in N$ **do**
    if $\alpha_n \in \widehat{s}_t \wedge target(n) \in \widehat{Act}_t$ **then**
      if $deon(n) = \mathcal{O}$ **then**
        Obligation Fulfilment
      else
        Prohibition Violation
      end if
    end if
    if $\neg\alpha_n \in \widehat{s}_t \wedge \alpha_n \in \widehat{s}_{t-1} \wedge \neg\beta_n \in \widehat{s}_t \wedge target(n) \in control(S)$ **then**
      if $deon(n) = \mathcal{F}$ **then**
        Prohibition Fulfilment
      else
        Obligation Violation
      end if
    end if
  end for
end while

Function 2 Resource Selection Algorithm

procedure RESOURCESELECTION(R, N, b)
  $S_1 \leftarrow \emptyset$
  while $\exists r \in R \setminus S_1 : cost(S \cup \{r\}) \leq b$ **do**
    $S_1 \leftarrow S_1 \cup \underset{r \in R \setminus S_1 : cost(S \cup \{r\}) \leq b}{\operatorname{argmin}} v(S \cup \{r\})$
  end while
  $S_2 \leftarrow \emptyset$
  while $\exists r \in R \setminus S_2 : cost(S \cup \{r\}) \leq b$ **do**
    $S_2 \leftarrow S_2 \cup \underset{r \in R \setminus S_2 : cost(S \cup \{r\}) \leq b}{\operatorname{argmin}} \frac{v(S \cup \{r\})}{cost(r)}$
  end while
  return $\underset{s \in \{S_1, S_2\}}{\arg\max}\, v(s)$
end procedure

5. Empirical Evaluation

In this section, we compare our *resource-bounded* monitor against tractable and intractable optimization approaches for selecting deployed resources. In particular, we compared our monitor against the following monitors:

- A *max-resources* monitor that maximizes the number of resources selected by selecting the resources with the lowest cost until no more resources can be selected according to the budget condition.
- A *dynamic* monitor using the algorithm proposed in (Criado & Such, 2016) to predict the actions of agents in each time step and select the resources to be deployed in each time step using dynamic programming according to the number of violations/fulfilments that are predicted to be observed through each resource. In particular, the *dynamic* monitor uses dynamic programming to select online the resources as in a 0-1 knapsack problem. To be able to use this dynamic programming algorithm the costs of the resources need to be integers (although the solution we propose on this paper does not require this constraint).
- An *optimal* monitor that obtains the best results in terms of the detection of norm violations and fulfilments at run-time. Note this monitor is intractable for reasonable-sized problems as it is obtained by testing exhaustively all resource sets satisfying the budget condition.

We have evaluated our proposal in a series of random experiments, which allow us to evaluate our proposal under a wide range of different situations and parameter values. For comparability reasons, we have also evaluated our proposal using benchmark problems.

5.1 Random Experiments

We implemented a simulator in which a set of agents perform actions in an environment randomly generated. We conducted experiments in which the number of resources R took a random value within the $[\![10, 50]\!]$ interval and the number of agents G is set to 100. Besides that, to be able to compare with the *optimal* monitor, we also considered small scenarios only, in which the number of resources R took a random value within the $[\![5, 10]\!]$ and the number of agents G is set to 5, as exploring all the subsets of resources is intractable for most of the cases with the default intervals.

The environment is defined as propositional symbols that are randomly initialised as being true or false. The number of propositions P takes a random value within the $[\![2R, 10R]\!]$ interval.

Actions allow agents to change the state of the environment. At the beginning of each simulation, a set of actions is randomly generated. For each action $\langle pre, post \rangle$ the elements pre and $post$ are defined as a set of consistent literals where each literal is defined over a randomly selected proposition from P which is negated with a 50% probability. To avoid that actions have too many constraints, which would be unrealistic and make actions to be only executed on a few situations making the monitoring problem over simplistic, the number of literals in pre and $post$ follows a Poisson[9] distribution with λ set to 1. The number of actions A takes a random value within the $[\![R, 2P]\!]$ interval (i.e., there is at maximum one action per each truth value of a proposition and at minimum one action per resource so there are no resources that do not control any action). Besides

9. Different distributions could be employed, we have just make use of a well-known distribution for generating natural numbers.

these actions, a NOP action, which has empty preconditions and postconditions, is created. In each step of the simulation, each agent selects randomly one action to execute.

Agents' actions are regulated by a randomly-created set of norms. For each norm $\langle deontic, target, activ, expir \rangle$ the elements are defined as follows: $deontic$ is randomly initialised with a deontic operator; $activ$ and $expir$ are defined as a set of consistent literals where each literal is defined over a randomly selected proposition from P which is negated with a 50% probability; and $action$ is randomly initialised with an action from A. Again, to allow that norms become relevant, the number of literals in $activ$ and $expir$ follows a Poisson distribution with λ set to 1. The number of norms takes random value within the $[\![A/2, A]\!]$ (i.e., there is at maximum one norm per each action).

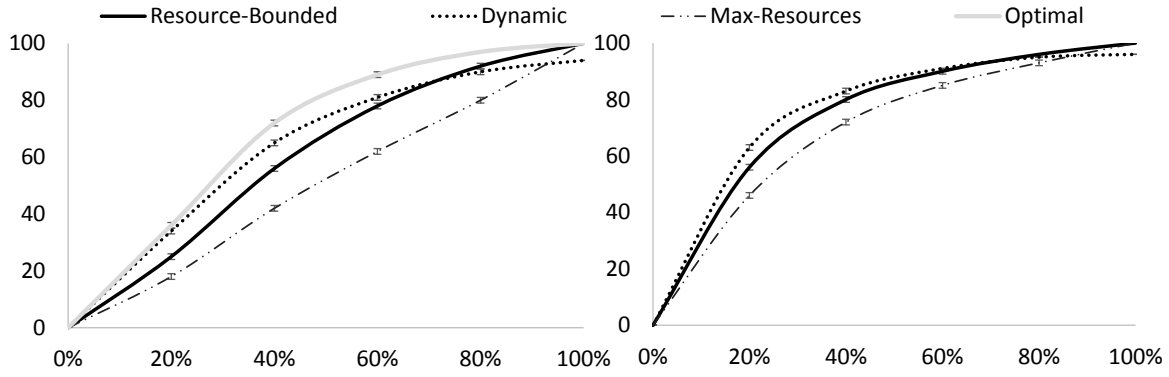
We randomly assign actions and propositions to resources. In particular, actions and propositions are uniformly distributed across resources with an overlap randomly defined within the interval $[0\%, 10\%]$. The cost of each resource is randomly defined as 1 plus a random noise generated by a Poisson distribution with λ depending on the size of the resource (i.e., the more elements in the control and sensor sets, the more expensive the resource is). Finally, we randomly set the budget ratio, which is the budget divided by the cost of all the resources in a simulation, to a random value within the $[0, 1]$ interval.

5.1.1 BUDGET

To analyse the performance of monitors with regard to their budget, we performed an experiment varying the budget ratio. In the experiment, we performed 1000 runs.

Table 1 shows the offline norm compliance analysis; i.e., the average percentage of norms that are perfectly and partially detectable per each type of monitor[10]. This analysis cannot be performed in case of the *dynamic* monitor as it selects a different set of resources in each time step and the detectability of norms varies at run-time. Note that the *resource-bounded* monitor maximises the percentage of detectable norms. However, this does not necessarily entails that it detects more violations and fulfilments at run-time, because some norms may never become relevant or some actions may never be executed during the simulations. This also explains why the percentage of detectable norms for the optimal monitor may be lower than for the other monitors (e.g., when the budget ratio is set to 100% the optimal monitor achieves on average a 63+21 percentage of detectable norms, whereas the resource-bounded and the max-resources achieve on average a 91+1 percentage of detectable norms). Recall that the optimal monitor is obtained by exhaustively trying all possible combinations of resources that satisfy the budget condition and selecting one of the combinations that achieve the best results online (not offline); i.e., that detects more violations and fulfilments. Note that there may be more than one set of resources that obtains the best results at run-time. For example, if the budget allows the optimal monitor to select all resources, then it has full observation capabilities and can detect all norm violations and fulfilments at run-time. However, note that due to the overlap between resources and to the fact that not all norms become relevant and not all actions are executed in the simulations, it is also possible that the norm monitor does not need to have full observation capabilities to detect all norm violations and fulfilments happening at run-time (i.e., it is possible there are other combinations of resources that also detect all norm violations and fulfilments). As what is observed through resources at run-time depends on the run-

10. For compactness, Table 1 does not include the 95% confidence intervals for the average percentage of norms that are perfectly and partially detectable. However, these intervals are lower or equal to 1% in all cases.



(a) $R \in [\![5, 10]\!]$ and $G = 5$

(b) $R \in [\![10, 50]\!]$ and $G = 100$

Figure 1: 95% Confidence Intervals for the Percentage of Detected Violations and Fulfilments (Y-axis) per Budget Ratio (X-axis)

time circumstances (i.e., location, agent behaviour, etc.), we analyse the run-time performance of the four monitors below.

Figure 1 shows the 95% confidence intervals for the percentage of detected violations and fulfilments at run-time for each monitor. Obviously, the detection of violations and fulfilments increases as the budget increases and more resources are deployed, which increases the information available to all monitors. Our *resource-bounded* monitor outperforms simple optimization approaches, offering on average a 7%-16% performance increase over a *max-resources* monitor. As can be observed in the figure, the confidence intervals obtained by our monitor and the *max-resources* monitor do not overlap (with the exception of scenarios with no observability and complete observability), which gives us an indication of the fact that these differences are significative. Indeed, a t-test ($\alpha < 0.05$) confirmed this hypothesis. Our monitor only suffers a 1%-4% decrease over a *dynamic* monitor, which, as next shown, is a costly solution that selects deployed resources in each time step. In this case, a t-test ($\alpha < 0.05$) failed to demonstrate that there are significant differences between the results obtained by our monitor and a *dynamic* monitor. When compared to an *optimal* monitor in small scenarios, our *resource-bounded* monitor is near optimal (it detects 89% of the violations and fulfilments detected by an *optimal* monitor). Note that the *optimal* monitor detects more violations and fulfilments at run-time since it has a better coverage of the norms that are actually violated and fulfilled, but it is intractable for reasonably sized problems.

Figure 2 shows the 95% confidence intervals for the time (in ms) needed by each norm monitor. This time includes the time needed for selecting deployed resources and the time needed for monitoring norm compliance. Given that the optimal monitor is obtained by exhaustively trying all resource sets satisfying the budget condition, this type of monitor does not select deployed resources and displaying its execution time makes no sense. The execution time of both *resource-bounded*, *max-resources* and *dynamic* monitors increases as size of the problem increases. However, this increase is significantly higher in case of the *dynamic* monitor, because predicting agent actions to adapt the resources to be deployed in each time step requires considerably more time. On average the *resource-bounded* monitor requires 71%-94% less time than a *dynamic* monitor, while obtaining a reasonably good performance in terms of the percentage of violations and fulfilments detected.

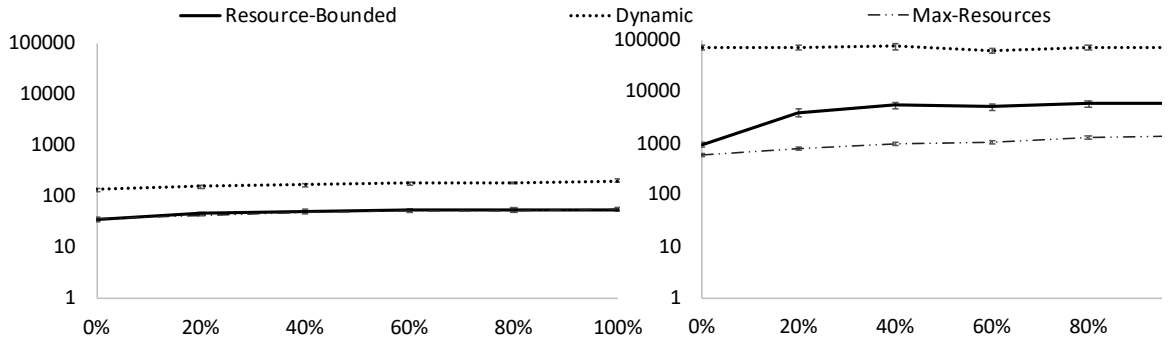
Budget Ratio	*Resource-bounded*	*Max-Resources*	*Optimal*
0%	0+0	0+0	0+0
20%	11+4	7+4	10+5
40%	24+16	20+12	23+17
60%	41+23	38+18	41+23
80%	62+21	61+18	50+24
100%	91+9	91+9	63+21

(a) $R \in [\![5, 10]\!]$ and $G = 5$

Budget Ratio	*Resource-bounded*	*Max-Resources*
0%	0+0	0+0
20%	30+14	26+13
40%	52+19	50+16
60%	67+18	67+15
80%	79+15	78+13
100%	89+11	89+11

(b) $R \in [\![10, 50]\!]$ and $G = 100$

Table 1: Average Percentage of Detectable Norms (Perfectly + Partially)



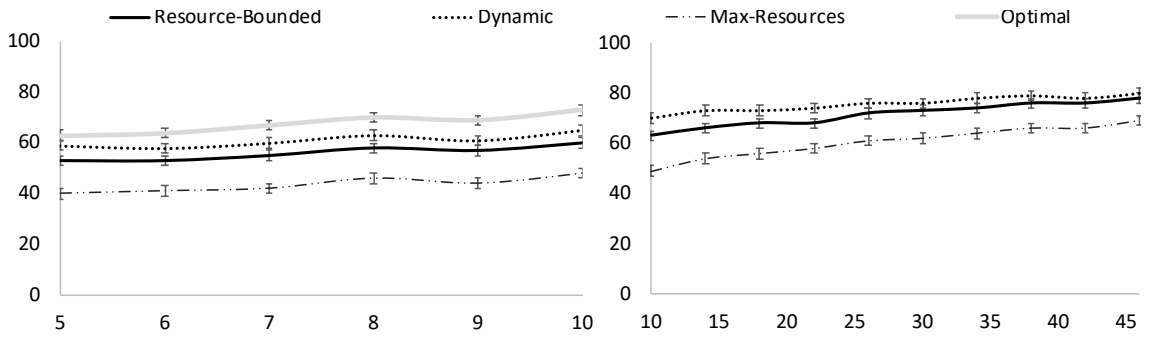
(a) $R \in [\![5, 10]\!]$ and $G = 5$

(b) $R \in [\![10, 50]\!]$ and $G = 100$

Figure 2: 95% Confidence Intervals for the Execution Time in ms and Logarithmic Scale (Y-axis) per Budget Ratio (X-axis)

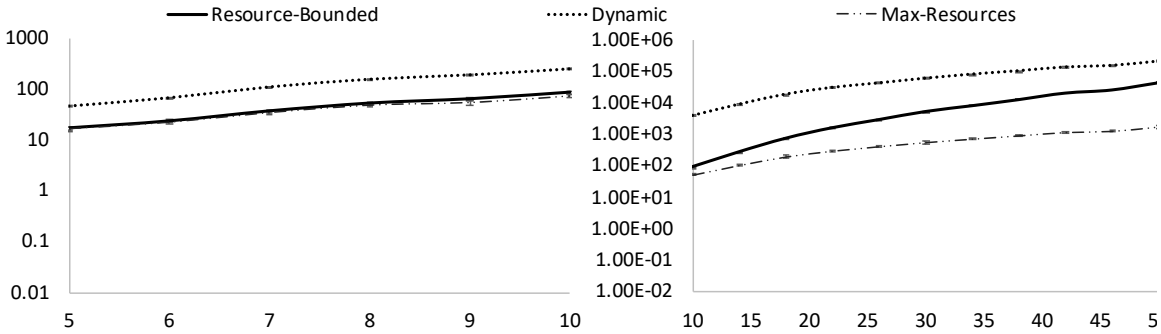
5.1.2 RESOURCES

To analyse the performance of monitors with regard to the number of resources, we performed different experiments varying the number of resources from 5 to 10 in small scenarios and from 10 to 50 in large scale scenarios. Again, we performed 1000 runs of the experiment. Note that this experiment is aimed at studying how well the different monitors scale with respect to an increasing



(a) $G = 5$ (b) $G = 100$

Figure 3: 95% Confidence Intervals for the Percentage of Detected Violations and Fulfilments (Y-axis) per number of Resources (X-axis)

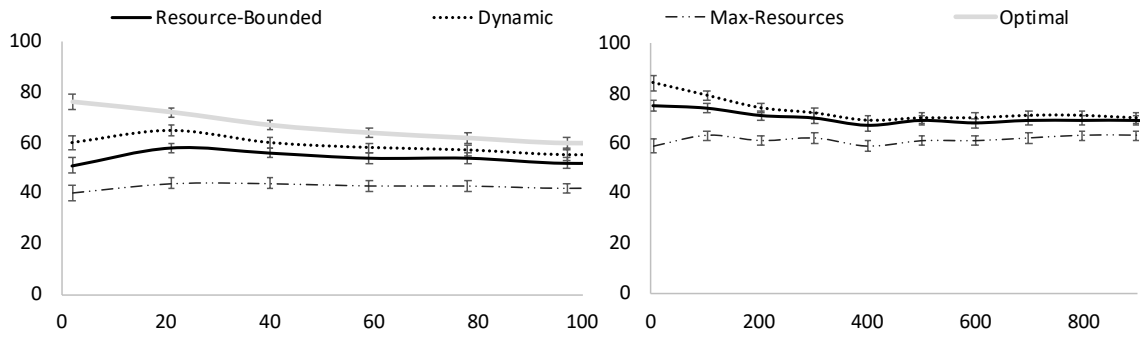


(a) $G = 5$ (b) $G = 100$

Figure 4: 95% Confidence Intervals for the Execution Time in ms and Logarithmic Scale (Y-axis) per number of Resources (X-axis)

number of resources. Hence, in the following we report the results obtained for the percentage of detected violations and fulfilments and the execution time.

Figure 3 shows the percentage of detected violations and fulfilments at run-time for each monitor. As the number of resources increases, the probability of overlaps between resources also increases, which entails that more violations and detections can be detected for all monitors. Note also that our *resource-bounded* monitor scales better with the number of resources; i.e., its performance becomes closer to the *dynamic* monitor as the number of resources increases. Figure 4 shows the time needed by each norm monitor. Obviously, the more resources, the more time is needed to select resources and to check norm compliance as more actions are observed. On average the *resource-bounded* monitor requires 66%-88% less time than a *dynamic* monitor, while it detects 92%-94% of the violations detected by a *dynamic* monitor.



(a) $R \in [\![5, 10]\!]$ and $G = 5$

(b) $R \in [\![10, 50]\!]$ and $G = 100$

Figure 5: 95% Confidence Intervals for the Percentage of Detected Violations and Fulfilments (Y-axis) per Number of Norms (X-axis)

5.1.3 NORMS

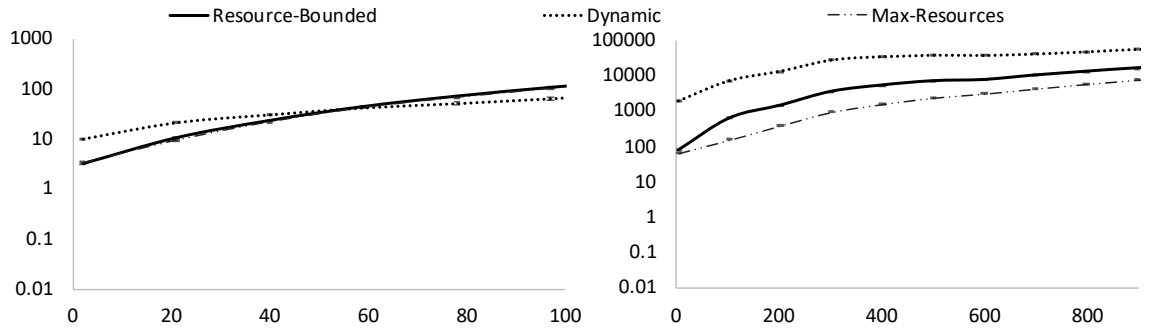
To analyse the performance of monitors with regard to the number of norms, we performed an experiment varying the number of norms from 2 to 100 in small scenarios and from 5 to 995 in large scale scenarios. Again, we performed 1000 runs of the experiment.

Figure 5 shows the percentage of detected violations and fulfilments at run-time for each monitor. As the number of norms increases, the difference between the *resource-bounded* and the *dynamic* monitor reduces. As the number of norms increases the number of norm violations and fulfilment increases and the almost any resource can detect a violation or fulfilment at any point in time. Regarding the executing time (see Figure 6), note the *dynamic* monitor is less affected by the number of norms and, as a result, its execution time in small scenarios remains quite stable regardless of the increase of norms. On the contrary, the execution time of *resource-bounded* and *max-resources* monitors depends on the number of norms and they may even need more time than a *dynamic* monitor to monitor norms. Note this only happens on small scenarios; in large scale scenarios, the increase on the number of agents and resources entails that the *dynamic* monitor is the monitor requiring significantly more time.

5.1.4 PROPOSITIONS

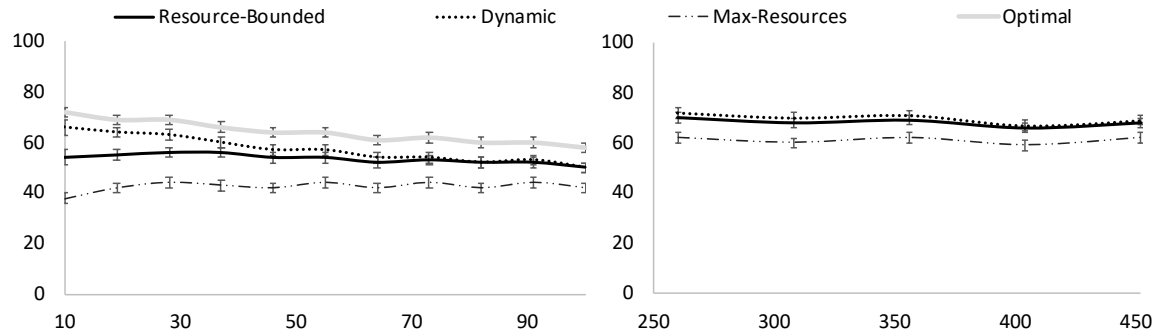
To analyse the performance of monitors with regard to the number of propositions, we performed an experiment varying the number of propositions from 10 to 100 in small scenarios and from 260 to 450 in large scale scenarios. Again, we performed 1000 runs of the experiment.

As the number of propositions increases, the difference between the *resource-bounded* and the *dynamic* monitor decreases and they obtain almost the same performance (see Figure 7). Regarding the execution time (see Figure 8), all monitors are affected by this factor in a similar way. On average, the *resource-bounded* monitor requires 68%-85% less time than a *dynamic* monitor, while it detects 93%-98% of the violations detected by a *dynamic* monitor.



(a) $R \in [\![5, 10]\!]$ and $G = 5$ (b) $R \in [\![10, 50]\!]$ and $G = 100$

Figure 6: 95% Confidence Intervals for the Execution Time in ms and Logarithmic Scale (Y-axis) per Number of Norms (X-axis)



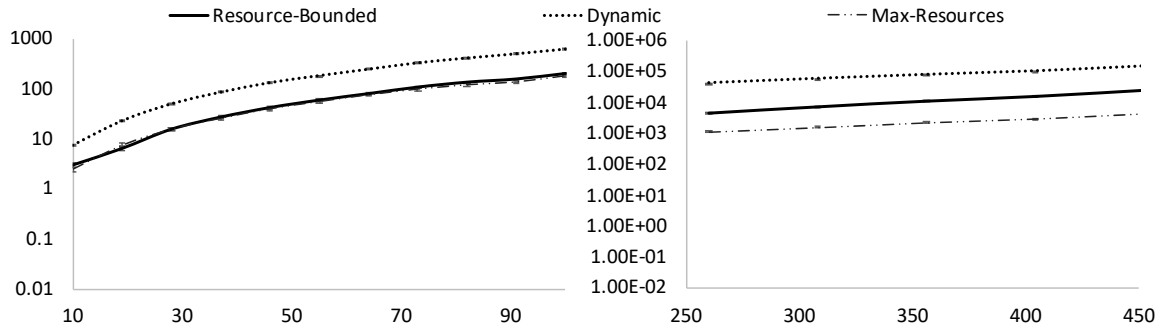
(a) $R \in [\![5, 10]\!]$ and $G = 5$ (b) $R \in [\![10, 50]\!]$ and $G = 100$

Figure 7: 95% Confidence Intervals for the Percentage of Detected Violations and Fulfilments (Y-axis) per Number of Propositions (X-axis)

5.1.5 ACTIONS

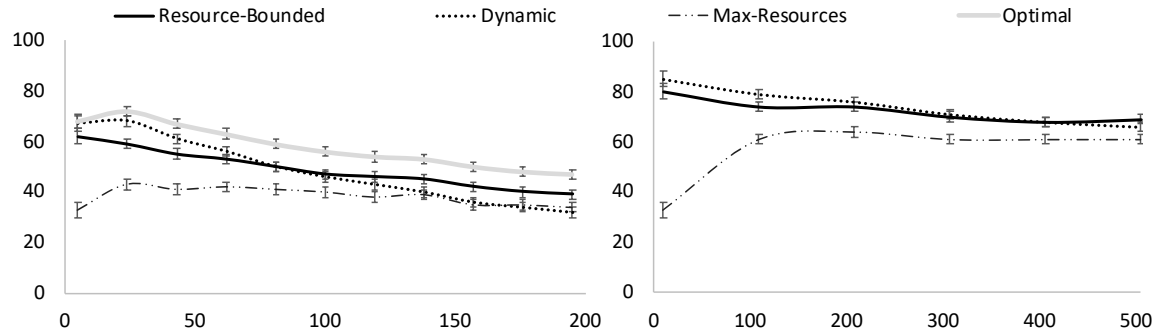
To analyse the performance of monitors with regard to the number of actions, we performed an experiment varying the number of actions from 5 to 195 in small scenarios and from 10 to 505 in large scale scenarios. Again, we performed 1000 runs of the experiment.

As the number of actions increases, the performance of the *dynamic* monitor deteriorates and *resource-bounded* detects more violations and fulfilments (see Figure 9). This is due to the fact that more action possibilities make impossible for the *dynamic* monitor to predict the next actions to be executed. Regarding the execution time (see Figure 8), the more actions the more time is needed to select deployed resources. The *resource-bounded* monitor requires 52%-90% less time than a *dynamic* monitor.



(a) $R \in [\![5, 10]\!]$ and $G = 5$ (b) $R \in [\![10, 50]\!]$ and $G = 100$

Figure 8: 95% Confidence Intervals for the Execution Time in ms and Logarithmic Scale (Y-axis) per Number of Propositions (X-axis)



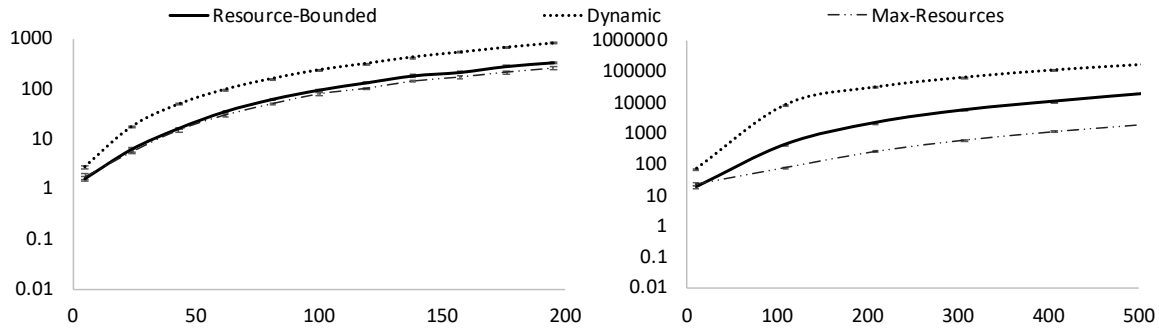
(a) $R \in [\![5, 10]\!]$ and $G = 5$ (b) $R \in [\![10, 50]\!]$ and $G = 100$

Figure 9: 95% Confidence Intervals for the Percentage of Detected Violations and Fulfilments (Y-axis) per Number of Actions (X-axis)

5.2 Benchmark Domains

To conduct a more comparable experimentation, we used four domains from a benchmark dataset provided by Ramírez and Geffner (2009), comprising of hundreds of problem descriptions. In particular, we used the domains: *blocks-words*, *easy-grid-navigation*, *intrusion-detection*, and *logistics*. These domains will allow future research on resource-bounded norm monitoring to compare against our approach and have been used in similar problems as plan recognition with limited observation capabilities (Keren et al., 2016), monitoring commitment abandonment (Pereira et al., 2017b, 2017a), and selecting norm monitoring placements with limited resources (Krzisch et al., 2017).

In each run, we randomly select one of the problem descriptions defined in the dataset and we initialise the set of actions and propositions as the actions and propositions in the selected problem description. In each time step of the simulation, an agent executes one action. In particular, the



(a) $R \in [\![5, 10]\!]$ and $G = 5$

(b) $R \in [\![10, 50]\!]$ and $G = 100$

Figure 10: 95% Confidence Intervals for the Execution Time in ms and Logarithmic Scale (Y-axis) per Number of Actions (X-axis)

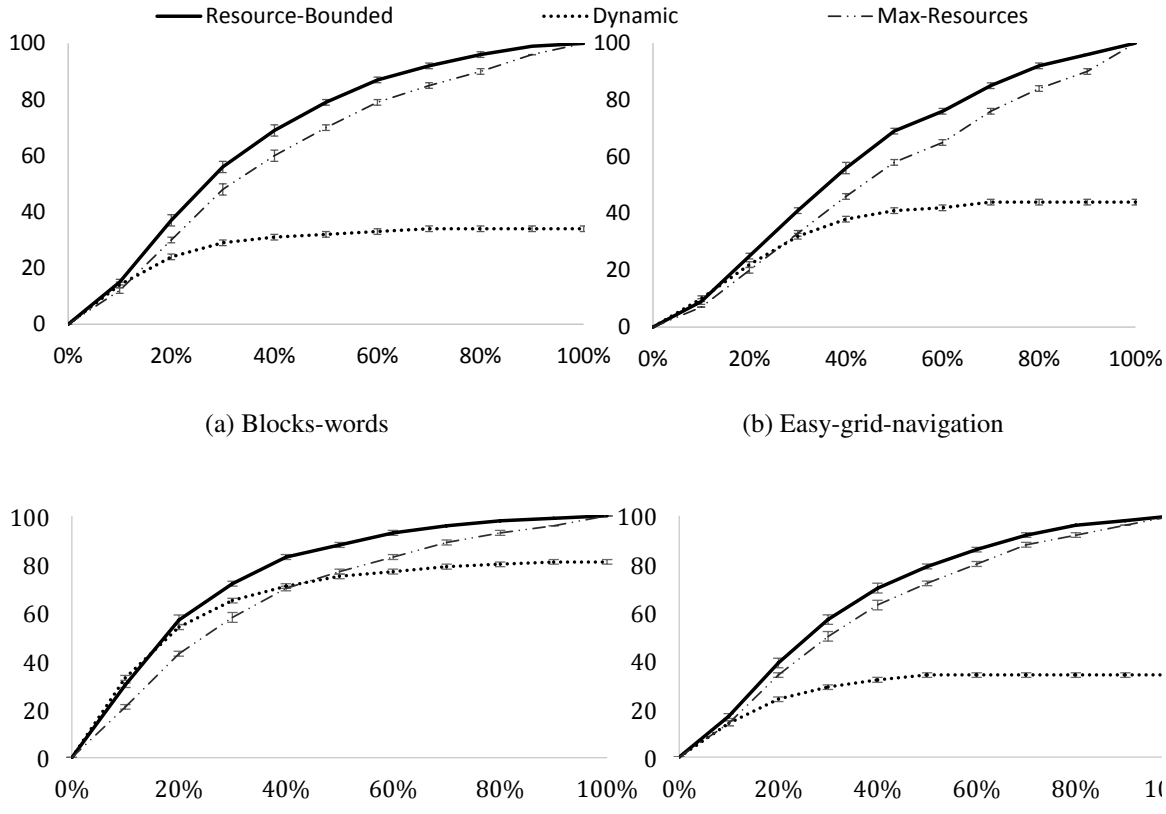
agent uses a Heuristic Search Planner to decide its actions and achieve the goal in the problem description[11].

Note that these problems do not contain any norm, so we randomly create a set of norms in each run. Again, each norm $\langle deontic, target, activ, expir, \rangle$ is defined as follows: $deontic$ is randomly initialised with a deontic operator; $activ$ and $expir$ are defined as a set of consistent literals where each literal is defined over a randomly selected proposition from the domain which is negated with a 50% probability; and $action$ is randomly initialised with an action from the domain. Again, to allow that norms become relevant, the number of literals in $activ$ and $expir$ follows a Poisson distribution with λ set to 1. The number of norms takes random value within the $[\![1, A]\!]$, where A is the number of actions in a problem domain.

We also create a set of resources by randomly assigning uniformly actions and propositions to resources with an overlap randomly defined within the interval $[0\%, 10\%]$. The cost of each resource is again defined as 1 plus a random noise generated by a Poisson distribution depending on the resource size. In each simulation the number of resources R took a random value within the $[\![10, A]\!]$ interval.

Note that time needed for checking norm compliance and the offline norm compliance analysis do depend on the particular actions taken by agents in each environment. Therefore, in the benchmark experiments we focus our attention on the online norm compliance analysis, as this process is the only one affected by the actions executed by agents that are determined by the problem description. Again to analyse the performance of monitors with regard to their budget, we varied budget ratio and conducted 1000 runs of each experiment. Figure 11 shows the 95% confidence intervals for the percentage of detected violations and fulfilments for each monitor in each domain. Our monitor obtains significant improvements over *dynamic* and *max-resources* monitors. Again, this was confirmed by t-tests with $\alpha < 0.05$. In particular, it obtains a 17%-114% improvement over a *dynamic* monitor and a 7%-12% improvement over a *max-resources* monitor. We can also observe that in three of these domains (namely *block-words*, *easy-grid-navigation* and *logistics*) the

11. The simulation is executed 50 steps, which ensures that the goal in the problem description is achieved. When the goal in the problem description has been achieved we randomly generate a new goal for that particular problem and situation.



(a) Blocks-words

(b) Easy-grid-navigation

(c) Intrusion-detection

(d) Logistics

Figure 11: 95% Confidence Intervals for the Percentage of Detected Violations and Fulfilments (Y-axis) per Budget Ratio (X-axis)

performance of the *dynamic* monitor is noticeably worse than that of the other two monitors. This is due to the fact that these domains are somewhat constrained (i.e., almost all actions are possible) and it is not possible for the *dynamic* monitor to make a precise prediction about the next action to be executed. Even when the budget is high and the *dynamic* monitor observes most of the actions and environment propositions, the *dynamic* monitor cannot detect the violation of obligations and the fulfilment of prohibitions as the selected resources change at run-time (i.e., it cannot be sure that the target of relevant norms has not been executed). On the contrary, our *resource-bounded* monitor shows a consistent good performance across all domains.

5.3 Summary

The conclusions of our evaluation are threefold:

1. Our *resource-bounded* monitor is more effective (i.e., detects more norm violations and fulfilments) than a simple resource maximization approach. Both in the benchmark and in the random experiments our monitor improved by a 7%-29% the detection of violations and fulfilments when compared to a *max-resources* monitor.

2. Our monitor is slighting less effective than an *optimal* monitor. In particular, our monitor detects an 83-89% of the violations and fulfilments detected by an *optimal* monitor. However, an optimal approach is intractable for reasonably sized problems.

3. Our monitor is better suited for practical applications than the *dynamic* monitor since: (i) the greedy approach is significantly more efficient than the dynamic approach, achieving a 77%-94% reduction in time for reasonable sized problems; (ii) our monitor offers guarantees about the norms that are detectable, whereas the *dynamic* monitor does not because selected resources are changed at run-time; (iii) our monitor obtains significantly better results in somehow constrained domains (achieving a 17%-117% improvement in the detection of norm violations and fulfilments), and similar results in over-constrained domains (suffering just a 1%-8% decrease in the detection of norm violations and fulfilments); and (iv) in many domains it is not feasible to change the resources monitored at run-time; e.g., in a smart home environment it is not possible/easy to change the monitored resources once they have been bought and installed.

6. Related Work

Our work has strong relations with norm representation, which have inspired our norm definition, and norm enforcement mechanisms, which encompasses the norm monitoring problem considered in this paper. In the following we briefly review the main literature in these two areas and position the contributions of this paper with respect to them.

6.1 Normative Systems

The term *normative system* was defined by Alchourrón and Bulygin (1971) as a set of statements in which there are some *normative statements* or norms.

Norms are usually formalised as specifications of deontic statements (defining an action as obligatory, forbidden, or permitted) aimed at regulating the life of software and/or human agents and the interactions among them (Rubino & Sartor, 2008). In particular, deontic logics (McNamara, 2014), which is the branch of symbolic logic focused on the formal analysis of norms and propositions about norms, is taken as a reference in many norm definitions. The standard version of deontic logics (von Wright, 1981) assumes that norms are coherent; i.e., that there are no conflicts between obligations, permissions and prohibitions. However, this assumption has been challenged by numerous authors aiming at representing and reasoning about legal and regulatory domains (Hansen et al., 2007; Dung & Sartor, 2011), giving rise to the development of variants of deontic logics using priorities (Hansen, 2008), norm hierarchies (Boella & van der Torre, 2003), etc. For the sake of clarity and simplicity, we base our norm definition on standard deontic logics and our running example does not include conflicting norms. However, our paper is agnostic with respect to the existence of conflicts, we do not constrain norms in a particular scenario to be conflict free (it is up to de system designer to define such requirement) and we have carried out experiments that include random norms that do not necessarily are conflict free. In scenarios with conflicting norms, our monitor will detect the violation and/or fulfilment of more than one norm for a single action executed, the decision of which norm prevails in this case is left to the norm enforcer entity, which will determine if a sanction or reward is applicable depending on the circumstances (Criado et al., 2016) or on the norm hierarchy (Boella & van der Torre, 2003).

Von Wright's classical approach also assumes a "closed legal system" in which obligations and prohibitions define exceptions to a default permission rule. Therefore, the notion of permission norm is not necessary, since a permission to execute an action is just the non-existence of any obligation or prohibition to perform that action. In this paper we also take this assumption as it is consistent with socio-technical systems. However, legal and regulatory domains are usually more complex (i.e., norm specifications are dynamic and formed by several normative systems), which has led to interesting questions as the need for weak (implicit or default) permissions and strong (explicit) permissions (Alchourrón & Bulygin, 1981), norm dynamics (Alchourrón & Bulygin, 1971) and defeasible logics (Governatori et al., 2013).

6.2 Norm Enforcement

Recent work has also addressed the problem of how to give a computational interpretation to norms so that they can be used in the design and execution of different types of systems (Criado et al., 2011). In the following we describe proposals on this topic classified into off-line and on-line norm enforcement.

6.2.1 OFF-LINE NORM ENFORCEMENT

The term off-line norm enforcement (also known as regimentation (Fornara & Colombetti, 2008a; Grossi et al., 2007)) covers works that are applied at design time to ensure that the resulting systems will satisfy the norms. These works can be further classified into mediation and compliance by design.

Mediation approaches (Bradshaw et al., 2003) consist in making the violation of norms impossible by controlling both the resources and the communication channel so that agents are prevented from deviating from ideal behaviour. For example, García-Camino et al. (2006) proposed a formalism based on rules for representing constraints on agent behaviours. This formalism is conceived as a "machine language" for implementing other higher level normative languages. As aforementioned, the regimentation of all actions can be not only difficult and/or costly, but also it may be inevitable or even preferable to allow agents to violate norms (Castelfranchi, 2003). The reasons behind desirability of norm violations are either that it is impossible to take a thorough control of all actions; or that it is possible to obtain higher benefits when norms are occasionally violated.

Compliance by design approaches (Sadiq et al., 2007) propose to incorporate norm compliance issues within systems design methodology. For example, Lu et al. (2007) have developed a method which to quantitatively measure the compliance degree of a given business process model against a set of control objectives or norms. Within the context of socio-technical systems, Kafali et al. (2016) proposed a revision tool (based on model checking) to ensure that specifications of socio-technical systems satisfy privacy norms. Our proposal, also performs some off-line analysis of norm compliance, but our aim is not to make the violation of norms impossible, but to determine which norms can be enforced on-line.

6.2.2 ON-LINE NORM ENFORCEMENT

The term on-line norm enforcement includes proposals on the enforcement of norms at run time. Given that our proposal falls into this category, this section reviews in more detail the most relevant control mechanisms that allow norms to have an effective influence on agent behaviours at run-time.

Most of the existing proposals on practical norm enforcement (Esteva et al., 2004; Cardoso & Oliveira, 2007; Meneguzzi et al., 2012; Vasconcelos et al., 2012; Gaertner et al., 2007) assume that monitors have unlimited resources to observe the actions performed by agents and the environment. In that case, checking norm compliance is easy as monitors have complete information about norm violations and fulfilments.

Exceptions to these complete monitoring approaches are four recent proposals (Bulling et al., 2013; Alechina et al., 2014; Criado & Such, 2017, 2016). Bulling et al. (2013) addressed the partial observability problem combining different norm monitors to build ideal monitors (i.e., monitors that together are able to detect the violation of a given set of norms). Alechina et al. (2014) proposed to synthesise an approximate set of norms that can be monitored given the observational capabilities of a monitor. However, there are circumstances in which norms cannot be modified (e.g., contract and law monitoring) or ideal monitors are expensive and/or not feasible. Criado and Such (2017) addressed partial observability by proposing methods to reconstruct the unobserved actions, but ignore the problem of resource selection and assume that a expert selects the resources deployed using some domain information. Only (Criado & Such, 2016) and (Krzisch et al., 2017) have addressed norm monitoring with limited resources. In particular, Criado and Such (2016) proposed to dynamically select the agents under surveillance by predicting agent actions (recall this is the *dynamic* monitor in our experiments in Section 5). However, we have demonstrated that Criado & Such's approach has some issues related to its efficiency, efficacy and other limitations constraining its practical applicability. Krzisch et al. (2017) proposed to select monitors based on just their costs or their observation capabilities. This approach has limitations in terms of formal results (i.e., no guarantees about the quality of the solution proposed are given), norm expressiveness, problem assumptions (e.g., agents cannot execute actions simultaneously), and experimental results (i.e., they do not compare the performance of their monitor at run time), which make impossible to compare against them in experiments.

Within the verification area, run-time verification has also tackled the problem of verifying that execution traces satisfy policies expressed as logical formulas (Leucker & Schallhart, 2009). Also in this area few approaches have tackled the real-world problem of dealing with incomplete information (or lossy traces, which is the term used in run-time verification literature). In particular, there has been a probabilistic approach (Stoller et al., 2012), that assigns probabilities to the events that could have occurred during an information gap and, based on these probabilities, calculates the probability that a policy has been violated. There are two main drawbacks with this approach: it requires a training set to calculate event probabilities, which in most cases is not available beforehand; and, more importantly, the detection of violations is not sound and monotonic (i.e., the later recovery of some unobserved events may change the verdict of the monitor). Joshi et al. (2017) have also proposed methods to identify monitorable policies (similarly to Alechina et al. (2014)), whereas Basin et al. (2012) have proposed use three-valued logics to monitor policies with lossy traces. However, none of these proposals has considered the question how to strategically decide which resources should be deployed so that the detection of policy violations over execution traces is optimal.

7. Conclusion

Norm monitors have applications in a wide range of information and communication technologies for sociotechnical systems. These applications range from network monitoring and abnormal detec-

tion tools for controlling security policies (Basin et al., 2010), to compliance monitors that check that actions of human and software agents in a given system comply by the privacy policies (Kafali et al., 2016). All these application areas could benefit from having advanced algorithms for run-time norm monitoring. In particular, the existing solutions (Alechina et al., 2013; Cardoso & Oliveira, 2007; Criado et al., 2013; Daskalopulu et al., 2002; Gaertner et al., 2007; Meneguzzi et al., 2012; Modgil et al., 2009; Leucker & Schallhart, 2009) for norm monitoring are ineffective for these application domains due to the inherent incompleteness of observations; e.g., network monitoring tools have limited resources to obtain information about the network, cannot monitor all events in the system and can be partially evaded by hackers. Even those solutions that try to overcome the partial observability problem cannot be applied to these domains as monitoring tools should be able to operate in real-time with limited resources and the policies cannot be modified, which entails that solutions ignoring norm monitoring costs (Criado & Such, 2016; Basin et al., 2012), solutions proposing to increase the monitoring resources (Bulling et al., 2013), and solutions proposing the adaptation of norms to what is observable (Alechina et al., 2014; Joshi et al., 2017) are all unsuitable for sociotechnical systems.

In this paper, we propose the first practical resource-bounded norm monitor, formally demonstrate its correctness, soundness, and study its complexity. We also evaluate empirically its effectiveness and efficiency; showing that our monitor achieves better results than other tractable optimization approaches and slightly worse results than intractable optimal approaches. The information model and algorithms proposed in this paper will be of interest to a wide range of practical applications within the socio-technical systems field. For example, monitoring is already used widely check compliance with security and privacy policies (specified as norms) in socio-technical systems. Besides that, the algorithms we propose for selecting deployed resources and monitoring norm compliance can be applied into other domains such as controlling autonomous systems, robotics and cyber-physical systems.

To simplify the presentation and focus on the essentials of our approach we assumed propositional formulas in the definition of actions and norms. Note this is a common assumption in the literature on norms and monitors (Alechina et al., 2014; Dastani et al., 2009; Havelund & Rosu, 2002). As future work we will consider enhanced representations defined using decidable fragments of first-order logic.

Appendix A. Complex Norms

Without loss of generality, in this paper we have considered simple action norms; i.e., norms that regulate the execution of a single action. However, the monitor proposed here can also be adapted to deal with more complex norms such as:

- *Attribute* norms that regulate properties of the world. Our monitor can accommodate these types of norms making the following changes:

  - Norm Definition. An attribute norm can be represented as a tuple $\langle deon, target, activ, expir\rangle$ where $deon, activ$ and $expir$ are defined as in Definition 4 and $target \in lit(P)$. Achievement obligations define properties that should be brought about at some point

while the norm is relevant, whereas maintenance prohibitions define properties that should not be brought about while the norm is relevant[12].

– Offline Norm Compliance. An attribute norm n is:
  * perfectly detectable when the following 3 conditions are satisfied:
    i) $target(n) \in lit(sensor(S))$,
    ii) $activ(n) \subseteq lit(sensor(S))$ or $activ(n) = \{\top\}$ or $activ(n) = \{\bot\}$.
    iii) $expir(n) \subseteq lit(sensor(S))$ or $expir(n) = \{\top\}$ or $expir(n) = \{\bot\}$.
  * partially detectable when the following 4 conditions are satisfied:
    i) n is not perfectly detectable.
    ii) $target(n) \in dis(S)$
    iii) $activ(n) \subseteq dis(S)$ or $activ(n) = \{\top\}$ or $activ(n) = \{\bot\}$.
    iv) $expir(n) \subseteq dis(S)$ or $expir(n) = \{\top\}$ or $expir(n) = \{\bot\}$.
  * non-detectable otherwise.

– Norm Compliance Information. The norm compliance information in case of attribute norms can be updated considering the literals that are known in each partial state as follows:

$$\left(\bigcup_{\substack{\forall n \in N : \alpha_n \in \widehat{s}_{t+1} \wedge \\ (\neg\alpha_n \in \widehat{s}_t \vee \\ (\alpha_n \in \widehat{s}_t \wedge target(n) \notin \widehat{s}_t \wedge \neg\beta_n \in \widehat{s}_t)}} \neg\beta_n \right) \bigcup \left(\bigcup_{\substack{\forall n \in N : \alpha_n \in \widehat{s}_{t+1} \wedge \alpha_n \in \widehat{s}_t \wedge \\ (target(n) \in \widehat{s}_t \vee \beta_n \in \widehat{s}_t)}} \beta_n \right)$$

– Norm Compliance Checking. Compliance with an attribute norm n is defined as follows:

$$\begin{cases} fulfilled & \text{iff } \alpha_n \in \widehat{s}_t \text{ and } deon(n) = \mathcal{O} \text{ and } target(n) \in \widehat{s}_t \\ fulfilled & \text{iff } \neg\alpha_n \in \widehat{s}_t \text{ and } \alpha_n \in \widehat{s}_{t-1} \text{ and } deon(n) = \mathcal{F} \text{ and } \neg\beta_n \in \widehat{s}_{t-1} \wedge \\ & target(n) \in lit(sensor(S)) \\ violated & \text{iff } \alpha_n \in \widehat{s}_t \text{ and } deon(n) = \mathcal{F} \text{ and } target(n) \in \widehat{s}_t \\ violated & \text{iff } \neg\alpha_n \in \widehat{s}_t \text{ and } \alpha_n \in \widehat{s}_{t-1} \text{ and } deon(n) = \mathcal{O} \text{ and } \neg\beta_n \in \widehat{s}_{t-1} \text{ and} \\ & target(n) \in lit(sensor(S)) \\ unknown & \text{otherwise} \end{cases}$$

– Resource Coverage. Given an attribute norm n, and a set of resources S; we define its coverage as:

$$\frac{\nmid \{target(n)\} \cap dis(S) \nmid + \nmid activ(n) \cap dis(S) \nmid + \nmid expir(n) \cap dis(S) \nmid}{1 + \nmid activ(n) \nmid + \nmid expir(n) \nmid}$$

Note that the theorems and proofs contained in this paper remain valid in case of attribute norms.

12. Note that a maintenance obligation stating that a property p should be maintained while the norm is relevant can be defined as a prohibition to maintain not p while the norm is relevant.

- *Conjunction norms* are norms that control the execution or achievement of a set of actions. Our monitor can accommodate them making the following changes:
  - Norm Definition. A conjunction norm is defined as a tuple $\langle deon, target, activ, expir\rangle$ where $deon, activ$ and $expir$ are defined as in Definition 4 and $target \subseteq A$. Thus, an obligation to execute a set of actions is interpreted as an obligation to execute all the actions in the target while the obligation is relevant. Similarly, a prohibition to execute a set for actions should be interpreted as a prohibition to execute all of the actions contained in the target while the prohibition is relevant[13].
  - Offline Norm Compliance. A conjunction norm n is:
    - perfectly detectable when the following conditions are satisfied:
      i) $activ(n) \subseteq lit(sensor(S)$ or $activ(n) = \{\top\}$ or $activ(n) = \{\bot\}$.
      ii) $expir(n) \subseteq lit(sensor(S))$ or $expir(n) = \{\top\}$ or $expir(n) = \{\bot\}$.
      iii) $target(n) \subseteq control(S)$.
    - partially detectable when the following conditions are satisfied:
      i) n is not perfectly detectable.
      ii) $activ(n) \subseteq dis(S)$ or $activ(n) = \{\top\}$ or $activ(n) = \{\bot\}$.
      iii) $expir(n) \subseteq dis(S)$ or $expir(n) = \{\top\}$ or $expir(n) = \{\bot\}$.
      iv) $target(n) \subseteq control(S)$.
    - non-detectable otherwise.
  - Norm Compliance Information. For each norm n, we define a set of propositions $\beta_{n_{a_i}}$ such that for each action a_i in $target(n)$, $\beta_{n_{a_i}}$ denotes that the execution of action a_i in the norm target has been observed while the norm was relevant. The norm compliance information in case of conjunction norms is updated as follows:

$$\left(\bigcup_{\substack{\forall n\in N:\alpha_n\in\widehat{s}_{t+1}\wedge\forall a_i\in target(n):\\(\neg\alpha_n\in\widehat{s}_t\vee\\(\alpha_n\in\widehat{s}_t\wedge a_i\notin\widehat{Act}_t\wedge\neg\beta_{n_{a_i}}\in\widehat{s}_t)}}\neg\beta_{n_{a_i}}\right)\bigcup\left(\bigcup_{\substack{\forall n\in N:\alpha_n\in\widehat{s}_{t+1}\wedge\forall a_i\in target(n):\\\alpha_n\in\widehat{s}_t\wedge(a_i\in\widehat{Act}_t\vee\beta_{n_{a_i}}\in\widehat{s}_t)}}\beta_{n_{a_i}}\right)$$

  - Norm Compliance Checking: compliance with a conjunction norm n is defined as follows:

$$\begin{cases} fulfilled & \text{iff } \alpha_n \in \widehat{s}_t \text{ and } deon(n) = \mathcal{O} \text{ and } \forall a_i \in target(n) : \beta_{a_i} \in \widehat{s}_t \\ fulfilled & \text{iff } \neg\alpha_n \in \widehat{s}_t \text{ and } \alpha_n \in \widehat{s}_{t-1} \text{ and } deon(n) = \mathcal{F} \text{ and} \\ & target(n) \subseteq control(S) \text{ and } \exists a_i \in target(n) : \neg\beta_{a_i} \in \widehat{s}_{t-1} \\ violated & \text{iff } \alpha_n \in \widehat{s}_t \text{ and } deon(n) = \mathcal{F} \text{ and } \forall a_i \in target(n) : \beta_{a_i} \in \widehat{s}_t \\ violated & \text{iff } \neg\alpha_n \in \widehat{s}_t \text{ and } \alpha_n \in \widehat{s}_{t-1} \text{ and } deon(n) = \mathcal{O} \\ & target(n) \subseteq control(S) \text{ and } \exists a_i \in target(n) : \neg\beta_{a_i} \in \widehat{s}_{t-1} \\ unknown & \text{otherwise} \end{cases}$$

13. Note disjunction norms can be rewritten as norms referring to a single action; e.g., $\langle \mathcal{F}, a \vee b, activ, expir\rangle \equiv \langle \mathcal{F}, a, activ, expir\rangle, \langle \mathcal{F}, b, activ, expir\rangle$.

– Resource Coverage: given a conjunction norm n, and a set of resources S; we define its coverage ($cover(S, n)$) as:

$$\frac{\nmid target(n) \cap control(S) \nmid + \nmid activ(n) \cap dis(S) \nmid + \nmid expir(n) \cap dis(S) \nmid}{\nmid target(n) \nmid + \nmid activ(n) \nmid + \nmid expir(n) \nmid}$$

Again, the theorems and proofs contained in this paper remain valid.

- *Mixed norms*, which target is defined over a set formed by actions and literals can also be supported by our monitor by combining the changes proposed for attribute and conjunction norms.

Appendix B. Norm Importance

In this paper we assumed that norms are of equal importance. However, it is also common that norms have different importance. This importance is related to the consequences of norm violations/fulfilments (e.g., the violation of some norms can be more severe than others) and is usually modelled by: predefined sanctions and rewards (Alechina et al., 2012), a preference relation over norms (Dignum et al., 2000), or a relative severity function (Gasparini et al., 2017).

The process by which our monitor checks compliance with norms does not depend on norm importance. However, to maximize the detection of violations and fulfilments of more important norms, our norm monitor can take norm importance into account when selecting the resources to be deployed. Here we will assume that norms are annotated with a number denoting their absolute or relative importance ($importance(n)$); i.e., the more important a norm is, the higher their importance. The norm monitor can account for the importance of norms in the resource value calculation so that more important norms contribute more than others when calculating the value of a set of resources:

$$v(S) = \frac{\sum_{n \in N} importance(n) \times cover(S, n)}{\sum_{n \in N} importance(n)}$$

Note that norm importance does not depend on the resources selected and our theorems and proofs remain valid.

Appendix C. Resource Failure

In this paper we assumed perfect resources that do not fault; i.e., a perfect resource always observes the execution of the actions that are controlled by it or the properties of the environment that are sensed by tit. However, it is also common that faulty resources cannot observe everything due to resource limited capabilities, defects, etc. For example, logging systems in networked environments may become temporary unavailable and unable to record actions happening in the network (Basin et al., 2012). The reliability of a resource is usually modelled by the probability of failure (Braband et al., 2009); i.e., likelihood that a piece of equipment will fail at a given time.

To maximize the detection of violations and fulfilments, our norm monitor can take resource failure into account when selecting the resources to be deployed. In particular, we will assume that any resource r is annotated with its probability of failure ($failure(n)$); i.e., $failure(n) > 0$ in case

of faulty resources; and $failure(n) = 0$ in case of non-faulty resources. The norm monitor can account for the probability of failure in the resource selection so that more faulty resources are less likely to be selected:

$$\max_{S \subseteq R} \prod_{r_i \in S} (1 - failure(r_i)) \times v(S) \textit{ subject to } cost(S) \leq b$$

Note also that the process by which our monitor checks compliance with norms assumes that resources are not faulty. Relaxing this assumption will entail that the monitor can no longer assume that some properties are invariant since some controlled actions (i.e., actions controlled by deployed resources) may have not been observed due to faults. In particular, the new partial state $\widehat{s}_{t+1}$ after the observation of some actions $\widehat{Act}_t$ is updated considering just the effect of actions $\widehat{Act}_t$ as follows:

$$\widehat{s}_{t+1} \leftarrow \widehat{s}_{t+1} \cup post(\widehat{Act}_t) \tag{7}$$

Resource failure also entails that no norm is perfectly detectable and that the detection of the violation of obligations and the fulfilment of permissions is no longer possible, since controlled actions may not be observed due to resource failure. In particular, the norm compliance check function should be redefined as:

$$\begin{cases} fulfilled & \text{iff } \alpha_n \in \widehat{s}_t \text{ and } deon(n) = \mathcal{O} \text{ and } target(n) \in \widehat{Act}_t \quad \text{(10a)} \\ violated & \text{iff } \alpha_n \in \widehat{s}_t \text{ and } deon(n) = \mathcal{F} \text{ and } target(n) \in \widehat{Act}_t \quad \text{(10c)} \\ unknown & \text{otherwise} \end{cases}$$

Faulty resources may lead to lower norm compliance detection rates, but note that our soundness and correctness results remain valid.

References

Alchourrón, C. E., & Bulygin, E. (1971). *Normative Systems*. Springer.

Alchourrón, C. E., & Bulygin, E. (1981). The expressive conception of norms. In *New studies in deontic logic*, pp. 95–124. Springer.

Alechina, N., Dastani, M., & Logan, B. (2012). Programming norm-aware agents. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1057–1064.

Alechina, N., Dastani, M., & Logan, B. (2013). Reasoning about normative update. In *Proc. of the International Joint Conference on Artificial Intelligence*, pp. 20–26.

Alechina, N., Dastani, M., & Logan, B. (2014). Norm approximation for imperfect monitors. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 117–124.

Basin, D. A., Klaedtke, F., Marinovic, S., & Zalinescu, E. (2012). Monitoring compliance policies over incomplete and disagreeing logs.. In *Proc. of the International Conference on Runtime Verification (RV)*, pp. 151–167. Springer.

Basin, D. A., Klaedtke, F., & Müller, S. (2010). Policy monitoring in first-order temporal logic.. In *Proc. of the International Conference on Computer-Aided Verification*, Vol. 6174, pp. 1–18. Springer.

Beheshti, R., Ali, A. M., & Sukthankar, G. R. (2015). Cognitive social learners: An architecture for modeling normative behavior.. In *Proc. of the AAAI Conference on Artificial Intelligence*, pp. 2017–2023.

Bloch, I. (1996). Information combination operators for data fusion: a comparative review with classification. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, *26*(1), 52–67.

Boella, G., & van der Torre, L. (2003). Permissions and obligations in hierarchical normative systems. In *Proc. of the International Conference on Artificial Intelligence and Law*, pp. 109–118. ACM.

Boutilier, C., & Brafman, R. I. (2001). Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, *14*(1), 105–136.

Braband, J., Vom Hövel, R., & Schäbe, H. (2009). Probability of failure on demand–the why and the how. *Computer Safety, Reliability, and Security*, 46–54.

Bradshaw, J., Uszok, A., Jeffers, R., Suri, N., Hayes, P., Burstein, M., Acquisti, A., Benyo, B., Breedy, M., Carvalho, M., et al. (2003). Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 835–842. ACM.

Bulling, N., Dastani, M., & Knobbout, M. (2013). Monitoring norm violations in multi-agent systems. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 491–498.

Cardoso, H., & Oliveira, E. (2007). Institutional reality and norms: Specifying and monitoring agent organizations. *International Journal of Cooperative Information Systems*, *16*(1), 67–95.

Castelfranchi, C. (2003). Formalising the informal? Dynamic social order, bottom-up social control, and spontaneous normative relations. *Journal of Applied Logic*, *1*(1-2), 47–92.

Criado, N., Argente, E., & Botti, V. (2011). Open issues for normative multi-agent systems. *AI Communications*, *24*(3), 233–264.

Criado, N. (2017). A practical resource-constrained norm monitor. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 1508–1510.

Criado, N., Argente, E., Noriega, P., & Botti, V. (2013). MaNEA: A distributed architecture for enforcing norms in open MAS. *Engineering Applications of Artificial Intelligence*, *26*(1), 76–95.

Criado, N., Black, E., & Luck, M. (2016). A coherence maximisation process for solving normative inconsistencies. *Autonomous Agents and Multi-Agent Systems*, *30*(4), 640–680.

Criado, N., & Such, J. M. (2016). Selective norm monitoring. In *Proc. of the International Joint Conference on Artificial Intelligence*, pp. 208–214.

Criado, N., & Such, J. M. (2017). Norm monitoring under partial action observability. *IEEE transactions on cybernetics*, *47*(2), 270–282.

Daskalopulu, A., Dimitrakos, T., & Maibaum, T. (2002). Evidence-based electronic contract performance monitoring. *Group Decision and Negotiation*, *11*(6), 469–485.

Dastani, M., Grossi, D., Meyer, J.-J. C., & Tinnemeier, N. (2009). Normative multi-agent programs and their logics. In *Knowledge Representation for Agents and Multi-Agent Systems*, pp. 16–31.

Dignum, F., Broersen, J., Dignum, V., & Meyer, J.-J. (2004). Meeting the deadline: Why, when and how. In *International Workshop on Formal Approaches to Agent-Based Systems*, pp. 30–40.

Dignum, F., Morley, D., Sonenberg, E. A., & Cavedon, L. (2000). Towards socially sophisticated BDI agents. In *Proc. of the International Conference on MultiAgent Systems*, pp. 111–118. IEEE.

Dung, P. M., & Sartor, G. (2011). The modular logic of private international law. *Artificial Intelligence and Law*, *19*(2-3), 233.

Esteva, M., Rosell, B., Rodríguez-Aguilar, J. A., & Arcos, J. L. (2004). AMELI: an agent-based middleware for electronic institutions. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 236–243.

Fornara, N., & Colombetti, M. (2008a). Specifying and enforcing norms in artificial institutions. In *Proc. of the international conference on Autonomous Agents and Multiagent Aystems (AAMAS)*, Vol. 3, pp. 1481–1484. ACM.

Fornara, N., & Colombetti, M. (2008b). Specifying and enforcing norms in artificial institutions. In *International Workshop on Declarative Agent Languages and Technologies*, pp. 1–17.

Gaertner, D., Garcia-Camino, A., Noriega, P., Rodriguez-Aguilar, J.-A., & Vasconcelos, W. (2007). Distributed norm management in regulated multiagent systems. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 624–631.

García-Camino, A., Rodríguez-Aguilar, J. A., Sierra, C., & Vasconcelos, W. (2006). A rule-based approach to norm-oriented programming of electronic institutions. *ACM SIGecom Exchanges*, *5*(5), 33–40.

Gasparini, L., Norman, T. J., & Kollingbaum, M. J. (2017). Severity-sensitive norm-governed multi-agent planning. *Autonomous Agents and Multi-Agent Systems*, 1–33.

Governatori, G., Olivieri, F., Rotolo, A., & Scannapieco, S. (2013). Computing strong and weak permissions in defeasible logic. *Journal of Philosophical Logic*, *42*(6), 799–829.

Grossi, D., Aldewereld, H., & Dignum, F. (2007). Ubi lex, ibi poena: Designing norm enforcement in e-institutions. In *Coordination, organizations, institutions and norms in agent systems II*, pp. 101–114.

Günay, A., Liu, Y., & Zhang, J. (2016). Promoca: Probabilistic modeling and analysis of agents in commitment protocols. *Journal of Artificial Intelligence Research*, *57*, 465–508.

Hansen, J. (2008). Prioritized conditional imperatives: problems and a new proposal. *Autonomous Agents and Multi-Agent Systems*, *17*(1), 11–35.

Hansen, J., Pigozzi, G., & Van Der Torre, L. (2007). Ten philosophical problems in deontic logic. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Havelund, K., & Rosu, G. (2002). Synthesizing monitors for safety properties. In *TACAS*, Vol. 2, pp. 342–356. Springer.

Iyer, R. K., & Bilmes, J. A. (2013). Submodular optimization with submodular cover and submodular knapsack constraints. In *Advances in Neural Information Processing Systems*, pp. 2436–2444.

Jones, A. J., Artikis, A., & Pitt, J. (2013). The design of intelligent socio-technical systems. *Artificial Intelligence Review*, *39*(1), 5–20.

Joshi, Y., Tchamgoue, G. M., & Fischmeister, S. (2017). Runtime verification of ltl on lossy traces. In *Proceedings of the Symposium on Applied Computing*, pp. 1379–1386. ACM.

Kafali, Ö., Ajmeri, N., & Singh, M. P. (2016). Revani: Revising and verifying normative specifications for privacy. *IEEE Intelligent Systems*, *31*(5), 8–15.

Keller, R. M. (1976). Formal verification of parallel programs. *Communications of the ACM*, *19*(7), 371–384.

Keren, S., Gal, A., & Karpas, E. (2016). Goal recognition design with non-observable actions. In *Proc. of the AAAI Conference on Artificial Intelligence*, pp. 3152–3158.

Kollingbaum, M. J., & Norman, T. J. (2002). Supervised interaction: creating a web of trust for contracting agents in electronic environments. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 272–279.

Krzisch, G., Oren, N., & Meneguzzi, F. (2017). Bounded-monitor placement in normative environments. In *Proceedings of the Workshop on Linked Democracy*. CEUR-WS.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., & Glance, N. (2007). Cost-effective outbreak detection in networks. In *Proc. of SIGKDD*, pp. 420–429.

Leucker, M., & Schallhart, C. (2009). A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, *78*(5), 293–303.

López y López, F., Luck, M., & d'Inverno, M. (2006). A normative framework for agent-based systems. *Computational & Mathematical Organization Theory*, *12*(2-3), 227–250.

Lorini, E. (2012). On the logical foundations of moral agency. In *Deontic Logic in Computer Science*, pp. 108–122.

Lu, R., Sadiq, S., & Governatori, G. (2007). Compliance aware business process design. In *International Conference on Business Process Management*, pp. 120–131. Springer.

McNamara, P. (2014). Deontic logic. In Zalta, E. N. (Ed.), *The Stanford Encyclopedia of Philosophy* (Winter 2014 edition). Metaphysics Research Lab, Stanford University.

Meneguzzi, F., Modgil, S., Oren, N., Miles, S., Luck, M., & Faci, N. (2012). Applying electronic contracting to the aerospace aftercare domain. *Engineering Applications of Artificial Intelligence*, *25*(7), 1471–1487.

Modgil, S., Faci, N., Meneguzzi, F., Oren, N., Miles, S., & Luck, M. (2009). A framework for monitoring agent-based normative systems. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 153–160.

Oren, N., Panagiotidi, S., Vázquez-Salceda, J., Modgil, S., Luck, M., & Miles, S. (2009). Towards a formalisation of electronic contracting environments. In *Coordination, organizations, institutions and norms in agent systems IV*, pp. 156–171.

Pereira, R. F., Oren, N., & Meneguzzi, F. (2017a). Detecting commitment abandonment by monitoring sub-optimal steps during plan execution. In *Proc. of the International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1685–1687. International Foundation for Autonomous Agents and Multiagent Systems.

Pereira, R. F., Oren, N., & Meneguzzi, F. (2017b). Monitoring plan optimality using landmarks and domain-independent heuristics. In *Proc. of the Workshop on Plan, Activity, and Intent Recognition*, pp. 867–873.

Ramírez, M., & Geffner, H. (2009). Plan recognition as planning. In *Proc. of the International Joint Conference on Artificial Intelligence*, pp. 1778–1783.

Reiter, R. (1978). On closed world data bases. In *Logic and data bases*, pp. 55–76. Springer.

Rubino, R., & Sartor, G. (2008). Preface. *Journal of Artificial Intelligence and Law*, *16*(1), 1–5.

Sadiq, S., Governatori, G., & Namiri, K. (2007). Modeling control objectives for business process compliance. In *Business process management*, pp. 149–164.

Singh, M. P. (2013). Norms as a basis for governing sociotechnical systems. *ACM Transactions on Intelligent Systems and Technology*, *5*(1), 21.

Stoller, S. D., Bartocci, E., Seyster, J., Grosu, R., Havelund, K., Smolka, S. A., & Zadok, E. (2012). Runtime verification with state estimation. In *Proc. of the International Conference on Runtime Verification (RV)*, pp. 193–207, Berlin, Heidelberg. Springer-Verlag.

Vasconcelos, W. W., García-Camino, A., Gaertner, D., Rodríguez-Aguilar, J. A., & Noriega, P. (2012). Distributed norm management for multi-agent systems. *Expert Systems with Applications*, *39*(5), 5990–5999.

Vasconcelos, W., Kollingbaum, M. J., & Norman, T. J. (2007). Resolving conflict and inconsistency in norm-regulated virtual organizations. In *Proc. of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 632–639. ACM.

von Wright, G. H. (1981). On the logic of norms and actions. In *New studies in deontic logic*, pp. 3–35. Springer.

Winikoff, M., & Cranefield, S. (2014). On the testability of BDI agent systems. *Journal of Artificial Intelligence Research*, *51*(1), 71–131.